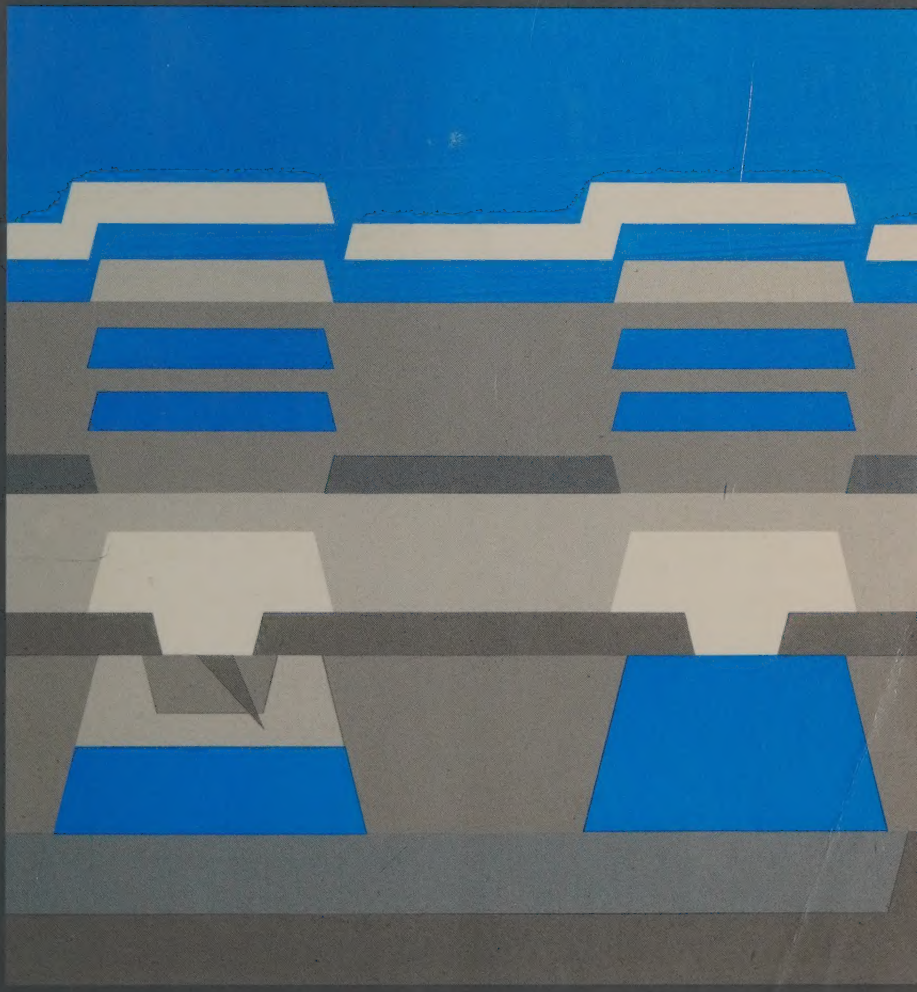
July, 1985

HITACHI

8/16-BIT MULTI-CHIP MICROCOMPUTER DATA BOOK



#U70

8/16-BIT MULTI-CHIP MICROCOMPUTER DATA BOOK

#U70

HITACHI

MEDICAL APPLICATIONS

Hitachi's products are not authorized for use in medical applications, including, but not limited to, use in life support devices without the written consent of the appropriate officer of Hitachi's sales company. Buyers of Hitachi's products are requested to notify Hitachi's sales offices when planning to use the products in the applications which involves medical applications.

When using this manual, the reader should keep the following in mind:

1. This manual may, wholly or partially, be subject to change without notice.
2. All rights reserved: No one is permitted to reproduce or duplicate, in any form, the whole or part of this manual without Hitachi's permission.
3. Hitachi will not be responsible for any damage to the user that may result from accidents or any other reasons during operation of his unit according to this manual.
4. This manual neither ensures the enforcement of any industrial properties or other rights, nor sanctions the enforcement right thereof.
5. Circuitry and other examples described herein are meant merely to indicate characteristics and performance of Hitachi semiconductor-applied products. Hitachi assumes no responsibility for any patent infringements or other problems resulting from applications based on the examples described herein.

July 1985
© Copyright 1985, Hitachi America Ltd.
Printed in U.S.A.

CONTENTS

■ **GENERAL INFORMATION**

- Quick Reference Guide 7
- Introduction of Packages 9
- Quality Assurance 18
- Reliability Test Data 24

■ **DATA SHEETS**

- **8-bit Multi-chip Microcomputers**

HD6803	Micro Processing Unit (NMOS)	33
HD6803-1	Micro Processing Unit (NMOS)	33
HD6303R	Micro Processing Unit (CMOS)	60
HD63A03R	Micro Processing Unit (CMOS)	60
HD63B03R	Micro Processing Unit (CMOS)	60
HD6303X	Micro Processing Unit (CMOS)	93
HD63A03X	Micro Processing Unit (CMOS)	93
HD63B03X	Micro Processing Unit (CMOS)	93
HD6303Y	Micro Processing Unit (CMOS)	129
HD63A03Y	Micro Processing Unit (CMOS)	129
HD63B03Y	Micro Processing Unit (CMOS)	129
HD6305X2	Micro Processing Unit (CMOS)	171
HD63A05X2	Micro Processing Unit (CMOS)	171
HD63B05X2	Micro Processing Unit (CMOS)	171
HD6305Y2	Micro Processing Unit (CMOS)	204
HD63A05Y2	Micro Processing Unit (CMOS)	204
HD63B05Y2	Micro Processing Unit (CMOS)	204
HD6800	Micro Processing Unit (NMOS)	237
HD68A00	Micro Processing Unit (NMOS)	237
HD68B00	Micro Processing Unit (NMOS)	237
HD6802	Microprocessor with Clock and RAM (NMOS)	269
HD6802W	Microprocessor with Clock and RAM (NMOS)	282
HD6809	Micro Processing Unit (NMOS)	295
HD68A09	Micro Processing Unit (NMOS)	295
HD68B09	Micro Processing Unit (NMOS)	295
HD6309	Micro Processing Unit (CMOS)	327
HD6809E	Micro Processing Unit (NMOS)	328
HD68A09E	Micro Processing Unit (NMOS)	328
HD68B09E	Micro Processing Unit (NMOS)	328
HD6309E	Micro Processing Unit (CMOS)	361
HD6821	Peripheral Interface Adapter (NMOS)	394
HD68A21	Peripheral Interface Adapter (NMOS)	394
HD68B21	Peripheral Interface Adapter (NMOS)	394
HD6321	Peripheral Interface Adapter (CMOS)	411
HD63A21	Peripheral Interface Adapter (CMOS)	411
HD63B21	Peripheral Interface Adapter (CMOS)	411
HD6840	Programmable Timer Module (NMOS)	430
HD68A40	Programmable Timer Module (NMOS)	430
HD68B40	Programmable Timer Module (NMOS)	430
HD6340	Programmable Timer Module (CMOS)	444
HD63A40	Programmable Timer Module (CMOS)	444
HD63B40	Programmable Timer Module (CMOS)	444
HD6843	Floppy Disk Controller (NMOS)	459
HD68A43	Floppy Disk Controller (NMOS)	459
HD6844	Direct Memory Access Controller (NMOS)	486
HD68A44	Direct Memory Access Controller (NMOS)	486
HD68B44	Direct Memory Access Controller (NMOS)	486
HD6845	CRT Controller (NMOS)	519
HD68A45	CRT Controller (NMOS)	519
HD68B45	CRT Controller (NMOS)	519
HD6845S	CRT Controller (NMOS)	550

HD68A45S CRT Controller (NMOS) ... 550
HD68B45S CRT Controller (NMOS) ... 550
HD6846 Combination ROM I/O Timer (NMOS) ... 591
HD6850 Asynchronous Communications Interface Adapter (NMOS) ... 612
HD68A50 Asynchronous Communications Interface Adapter (NMOS) ... 612
HD6350 Asynchronous Communications Interface Adapter (CMOS) ... 623
HD63A50 Asynchronous Communications Interface Adapter (CMOS) ... 623
HD63B50 Asynchronous Communications Interface Adapter (CMOS) ... 623
HD6852 Synchronous Serial Data Adapter (NMOS) ... 634
HD68A52 Synchronous Serial Data Adapter (NMOS) ... 634
HD46508 Analog Data Acquisition Unit (NMOS) ... 648
HD46508-1 Analog Data Acquisition Unit (NMOS) ... 648
HD46508A Analog Data Acquisition Unit (NMOS) ... 648
HD46508A-1 Analog Data Acquisition Unit (NMOS) ... 648
HD146818 Real Time Clock Plus RAM (CMOS) ... 668

- **16-bit Multi-chip Microcomputer**

HD68000-6 Micro Processing Unit (NMOS) ... 689
HD68000-8 Micro Processing Unit (NMOS) ... 689
HD68000-10 Micro Processing Unit (NMOS) ... 689
HD68000-12 Micro Processing Unit (NMOS) ... 689
HD68000Y6 Micro Processing Unit (NMOS) ... 689
HD68000Y8 Micro Processing Unit (NMOS) ... 689
HD68000Y10 Micro Processing Unit (NMOS) ... 689
HD68000Y12 Micro Processing Unit (NMOS) ... 689
HD68000P6 Micro Processing Unit (NMOS) ... 689
HD68000P8 Micro Processing Unit (NMOS) ... 689
HD68000PS6 Micro Processing Unit (NMOS) ... 689
HD68000PS8 Micro Processing Unit (NMOS) ... 689
HD68450-4 Direct Memory Access Controller (NMOS) ... 769
HD68450-6 Direct Memory Access Controller (NMOS) ... 769
HD68450-8 Direct Memory Access Controller (NMOS) ... 769
HD68450-10 Direct Memory Access Controller (NMOS) ... 769
HD68450Y4 Direct Memory Access Controller (NMOS) ... 769
HD68450Y6 Direct Memory Access Controller (NMOS) ... 769
HD68450Y8 Direct Memory Access Controller (NMOS) ... 769
HD68450Y10 Direct Memory Access Controller (NMOS) ... 769
HD63463-4 Hard Disk Controller (CMOS) ... 817
HD63463-6 Hard Disk Controller (CMOS) ... 817
HD63463-8 Hard Disk Controller (CMOS) ... 817
HD63484-4 Advanced CRT Controller (CMOS) ... 818
HD63484-6 Advanced CRT Controller (CMOS) ... 818
HD63484-8 Advanced CRT Controller (CMOS) ... 818

■ **INTRODUCTION OF THE RELATED DEVICES**

- 4-bit Single-chip Microcomputer ... 877
- 8-bit Single-chip Microcomputer ... 881
- IC Memory ... 890
- LCD Driver Series ... 892
- Gate Array ... 894
- CODEC/Filter Combo LSI ... 896
- Speech Synthesis LSI ... 896

■ **HITACHI SALES OFFICE LOCATIONS** ... 897

GENERAL INFORMATION

- Quick Reference Guide
- Introduction of Packages
- Quality Assurance
- Reliability Test Data

QUICK REFERENCE GUIDE

■ 8-BIT MULTI-CHIP MICROCOMPUTER

Division		Type No.	Old Type No.	LSI Characteristics: Process	Clock Frequency (MHz)	Supply Voltage (V)	Operating*** Temperature (°C)	Package†	Function	Compatibility	Reference Page
MPU		HD6803		NMOS	1.0	5.0	0 ~ +70	DP-40	Microprocessor +128 Bytes of RAM	MC6803	33
		HD6803-1			1.25					MC6803-1	33
		HD6303R		CMOS	1.0	5.0	0 ~ +70	DP-40 FP-54 CG-40	Microprocessor +128 Bytes of RAM		60
		HD63A03R			1.5						60
		HD63B03R			2.0						60
		HD6303X		CMOS	1.0	5.0	0 ~ +70	DP-64S FP-80	Microprocessor +192 Bytes of RAM		93
		HD63A03X			1.5						93
		HD63B03X			2.0						93
		HD6303Y		CMOS	1.0	5.0	0 ~ +70	DP-64S	Microprocessor +256 Bytes of RAM		129
		HD63A03Y			1.5						129
		HD63B03Y			2.0						129
		HD6305X2		CMOS	1.0	5.0	0 ~ +70	DP-64S FP-64	Microprocessor +128 Bytes of RAM		171
		HD63A05X2			1.5						171
		HD63B05X2			2.0						171
		HD6305Y2		CMOS	1.0	5.0	0 ~ +70	DP-64S FP-64	Microprocessor +256 Bytes of RAM		204
		HD63A05Y2			1.5						204
		HD63B05Y2			2.0						204
		HD6800	HD46800D	NMOS	1.0	5.0	-20 ~ +75	DP-40	Microprocessor	MC6800	237
		HD68A00	HD468A00		1.5					MC68A00	237
		HD68B00	HD468B00		2.0					MC68B00	237
		HD6802	HD46802	NMOS	1.0	5.0	-20 ~ +75	DP-40	Microprocessor+Clock+128 Bytes of RAM	MC6802	269
		HD6802W		NMOS	1.0	5.0	-20 ~ +75	DP-40	Microprocessor+Clock+256 Bytes of RAM		282
		HD6809		NMOS	1.0	5.0	-20 ~ +75	DP-40	High-End 8-Bit Microprocessor	MC6809	295
		HD68A09			1.5					MC68A09	295
		HD68B09			2.0					MC68B09	295
		HD6309**		CMOS	2.0	5.0	-20 ~ +75	DP-40	High-End 8-Bit Microprocessor		327
					2.5						327
					3.0						327
		HD6809E		NMOS	1.0	5.0	-20 ~ +75	DP-40	High-End 8-Bit Microprocessor (External Clock Type)	MC6809E	328
		HD68A09E			1.5					MC68A09E	328
		HD68B09E			2.0					MC68B09E	328
		HD6309E*		CMOS	2.0	5.0	-20 ~ +75	DP-40	High-End 8-Bit Microprocessor (External Clock Type)		361
					2.5						361
					3.0						361
Peripheral LSI	PIA	HD6821	HD46821	NMOS	1.0	5.0	-20 ~ +75	DP-40	Peripheral Interface Adapter	MC6821	394
		HD68A21	HD468A21		1.5					MC68A21	394
		HD68B21	HD468B21		2.0					MC68B21	394
		HD6321*		CMOS	1.0	5.0	-20 ~ +75	DP-40 FP-54	Peripheral Interface Adapter		411
		HD63A21*			1.5						411
		HD63B21*			2.0						411
	PTM	HD6840		NMOS	1.0	5.0	-20 ~ +75	DP-28	Programmable Timer Module	MC6840	430
		HD68A40			1.5					MC68A40	430
		HD68B40			2.0					MC68B40	430
		HD6340*		CMOS	1.0	5.0	-20 ~ +75	DP-28	Programmable Timer Module		444
		HD63A40*			1.5						444
		HD63B40*			2.0						444
	FDC	HD6843	HD46503S	NMOS	1.0	5.0	0 ~ +75	DP-40	Floppy Disk Controller	MC6843	459
		HD68A43	HD46503S-1		1.5						459
	DMAC	HD6844	HD46504	NMOS	1.0	5.0	-20 ~ +75	DP-40	Direct Memory Access Controller	MC6844	486
		HD68A44	HD46504-1		1.5					MC68A44	487
		HD68B44	HD46504-2		2.0					MC68B44	488
	CRTC	HD6845	HD46505R	NMOS	1.0	5.0	-20 ~ +75	DP-40	CRT Controller (3.0MHz High-speed Display)	MC6845	519
		HD68A45	HD46505R-1		1.5					MC68A45	519
		HD68B45	HD46505R-2		2.0					MC68B45	519
		HD6845S	HD46505S	NMOS	1.0	5.0	-20 ~ +75	DP-40	CRT Controller (3.7MHz High-speed Display)		550
		HD68A45S	HD46505S-1		1.5						550
		HD68B45S	HD46505S-2		2.0						550
	COMBO	HD6846	HD46846	NMOS	1.0	5.0	-20 ~ +75	DP-40	Combination ROM I/O Timer	MC6846	591
	ACIA	HD6850	HD46850	NMOS	1.0	5.0	-20 ~ +75	DP-24	Asynchronous Communications Interface Adapter	MC6850	612
		HD68A50	HD468A50		1.5					MC68A50	612
		HD6350		CMOS	1.0	5.0	-20 ~ +75	DP-24	Asynchronous Communications Interface Adapter		623
		HD63A50			1.5						623
		HD63B50			2.0						623
	SSDA	HD6852	HD46852	NMOS	1.0	5.0	-20 ~ +75	DP-24	Synchronous Serial Data Adapter	MC6852	634
		HD68A52	HD468A52		1.5					MC68A52	634
	ADU	HD46508		NMOS	1.0	5.0	-20 ~ +75	DP-40	Analog Data Acquisition Unit		648
		HD46508-1			1.5						648
		HD46508A			1.0						648
		HD46508A-1			1.5						648
	RTC	HD146818		CMOS	1.0	5.0	0 ~ +70	DP-24 FP-24	Real Time Clock Plus RAM	MC146818	668

* Preliminary ** Under development *** Wide Temperature Range (-40 ~ +85°C) version is available.
† DP: Plastic DIP, FP: Plastic Flat Package, CG: Glass-sealed Ceramic Leadless Chip Carrier

■ 16-BIT MULTI-CHIP MICROCOMPUTER

Division		Type No.	LSI Characteristics					Function	Compatibility	Reference Page
			Process	Clock Frequency (MHz)	Supply Voltage (V)	Operating Temperature (°C)	† Package			
MPU		HD68000-6	NMOS	6	5.0	0 ~ +70	DC-64	Microprocessor	MC68000L6	689
		HD68000-8		8					MC68000L8	689
		HD68000-10		10					MC68000L10	689
		HD68000-12		12.5					MC68000L12	689
		HD68000Y6		6			PGA-68		MC68000R6	689
		HD68000Y8		8					MC68000R8	689
		HD68000Y10		10					MC68000R10	689
		HD68000Y12		12.5					MC68000R12	689
		HD68000P6		6			DP-64		MC68000G6	689
		HD68000P8		8					MC68000G8	689
		HD68000PS6		6			DP-64S		—	689
		HD68000PS8		8					—	689
Peripheral LSI	DMAC	HD68450-4	NMOS	4	5.0	0 ~ +70	DC-64	Direct Memory Access Controller	MC68450L4	769
		HD68450-6		6					MC68450L6	769
		HD68450-8		8					MC68450L8	769
		HD68450-10		10					MC68450L10	769
		HD68450Y4		4			PGA-68		—	769
		HD68450Y6		6					—	769
		HD68450Y8		8					—	769
		HD68450Y10		10					—	769
	HDC	HD63463-4**	CMOS	4	5.0	0 ~ +70	DC-48	Hard Disk Controller	—	817
		HD63463-6**		6					—	817
		HD63463-8**		8					—	817
	ACRTC	HD63484-4*	CMOS	4	5.0	0 ~ +70	DC-64	Advanced CRT Controller	—	818
		HD63484-6*		6					—	818
		HD63484-8*		8					—	818

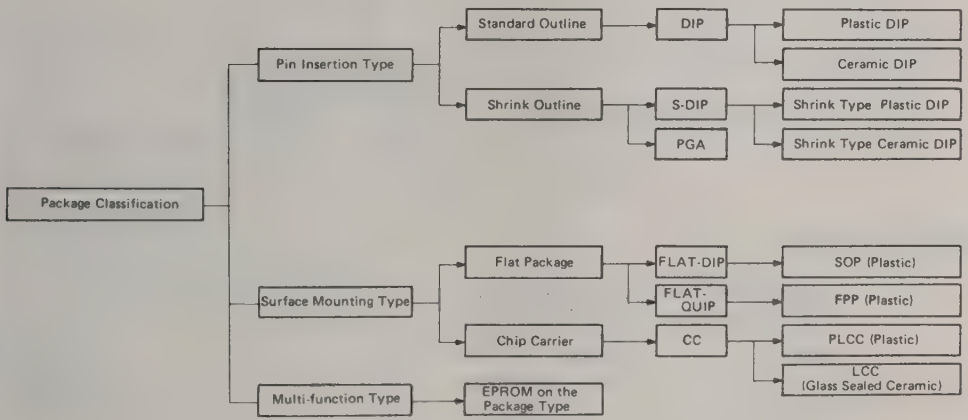
* Preliminary ** Under development † DP; Plastic DIP, DC; Ceramic DIP, PGA; Pin Grid Array

INTRODUCTION OF PACKAGES

Hitachi microcomputer devices are offered in a variety of packages, to meet various user requirements.

1. Package Classification

When selecting suitable packaging, please refer to the Package Classifications given in Fig. 1 for pin insertion, surface mount, and multi-function types, in plastic and ceramic.



DIP; DUAL IN LINE PACKAGE
S-DIP; SHRINK DUAL IN LINE PACKAGE
PGA: PIN GRID ARRAY
FLAT-DIP; FLAT DUAL IN LINE PACKAGE
FLAT-QUIP; FLAT QUAD IN LINE PACKAGE
CC: CHIP CARRIER
SOP; SMALL OUTLINE PACKAGE
FPP; FLAT PLASTIC PACKAGE
PLCC; PLASTIC LEADED CHIP CARRIER
LCC ; LEADLESS CHIP CARRIER

Fig. 1 Package Classification according to Material and Printed Circuit Board Mounting Type

2. Type No. and Package Code Indication

Type No. of Hitachi multi-chip microcomputer device is followed by package material and outline specifications, as shown below. The package type used for each device is identified by code as follows, illustrated in the data sheet of each device.

When ordering, please write the package code beside the type number.

Type No. Indication

HD××××P

Package Classification	
No Indication	; Ceramic DIP
P	; Plastic DIP
F (FP)	; SOP, FPP
CG	; LCC
Y	; PGA (16-bit microcomputer device)

(Note) The HD68000 with shrink type plastic DIP (DP-64S) has a different type No. from other devices.

Type No.; HD68000PS8

Package designation

Package Code Indication

DP—64S

Outline	Materials	Number of Pins	Additional Outline
D ; DIP C ; CC F ; FLAT	P ; Plastic G ; Glass Sealed ceramic C ; Ceramic		S; S-DIP

(Note) PGA packages of 16-bit microcomputer devices have a different indication.

Package Code Indication; PGA-68

Package Classification | Number of Pins

3. Package Dimensional Outline

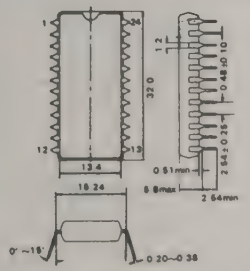
Hitachi multi-chip microcomputer device employs the packages shown in Table 1 according to the mounting method on the PCB.

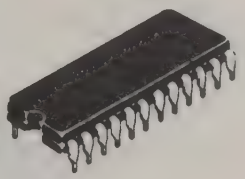
Table 1 Package List

Method of Mounting	Package Classification		Package Material	Package Code
Pin Insertion Type	Standard Outline (DIP)		Plastic	DP-24 DP-28 DP-40 DP-64
			Ceramic	DC-48 DC-64
	Shrink Outline	S-DIP	Plastic	DP-64S
		PGA	Glass Sealed Ceramic	PGA-68
Surface Mounting Type	Flat Package	FLAT-DIP (SOP)	Plastic	FP-24
		FLAT-QUIP (FPP)		FP-54 FP-64 FP-80
	Chip Carrier (LCC)		Glass Sealed Ceramic	CG-40

Plastic DIP

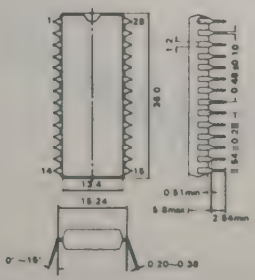
- **DP-24**

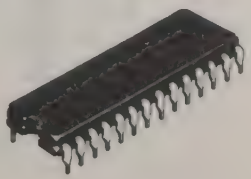




(Unit: mm)

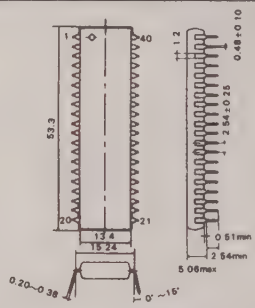
- **DP-28**

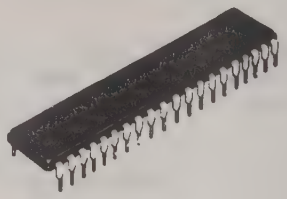




(Unit: mm)

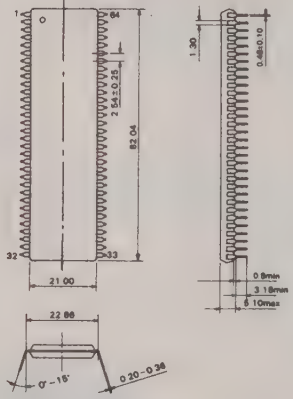
- **DP-40**





(Unit: mm)

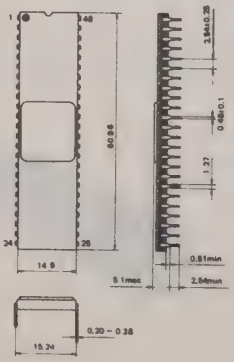
- **DP-64**

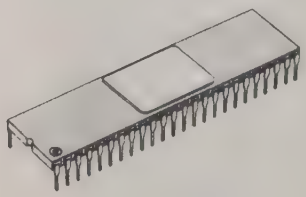


(Unit: mm)

Ceramic DIP

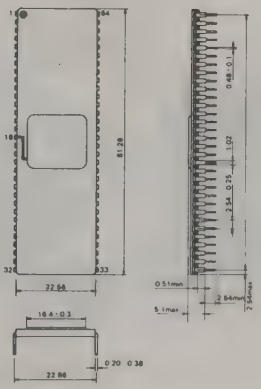
- **DC-48**

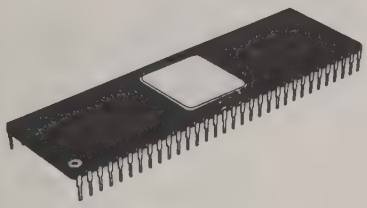




(Unit: mm)

- **DC-64**

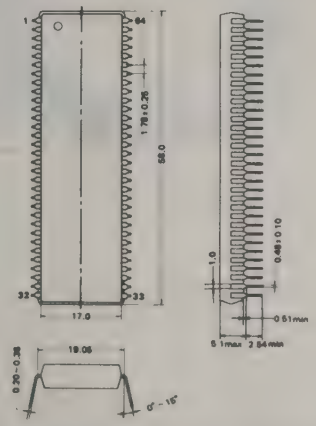


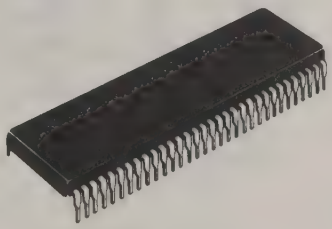


(Unit: mm)

Shrink Type Plastic DIP

- **DP-64S**

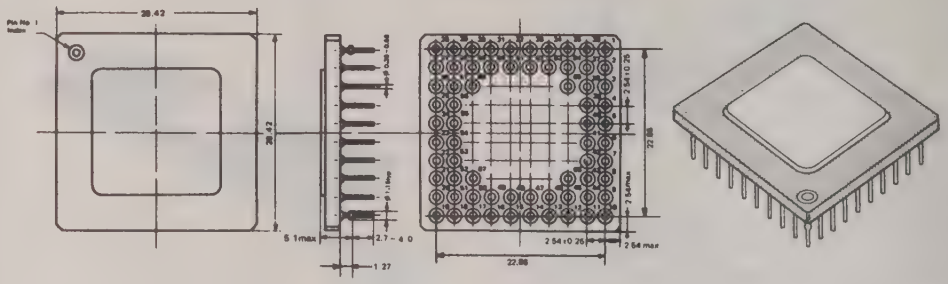




(Unit: mm)

Pin Grid Array

- PGA-68

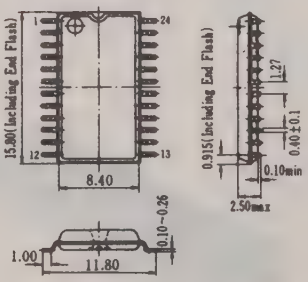


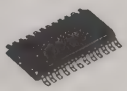
(Unit: mm)

Flat Package

< SOP >

- FP-24

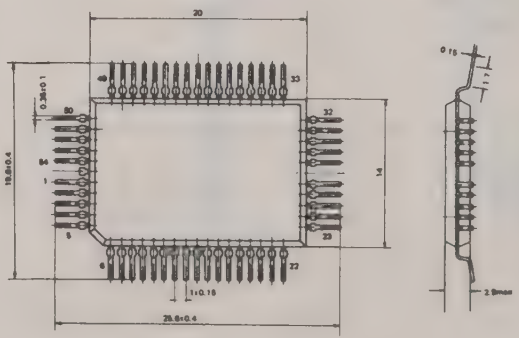


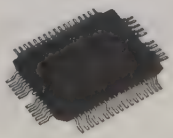


(Unit: mm)

< FPP >

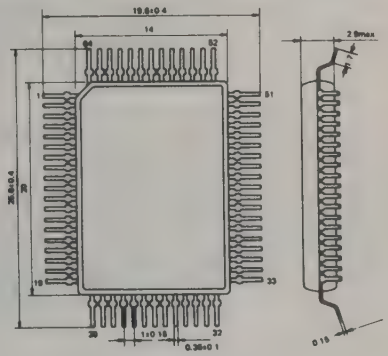
- FP-54

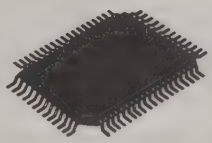




(Unit: mm)

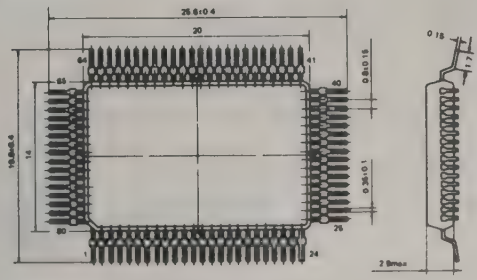
< FPP >
● FP-64

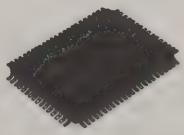




(Unit: mm)

< FPP >
● FP-80

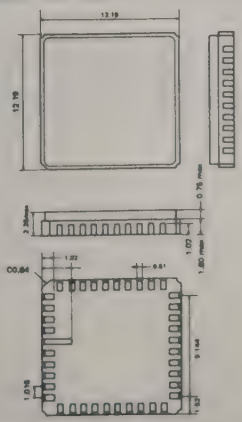




(Unit: mm)

Leadless Chip Carrier

● CG-40



(Unit: mm)

4. Mounting Method

Package lead pins are surface treated with solder coating or plating to facilitate PCB mounting. The lead pins are connected to the package by eutectic solder. Common connecting method of leads and precautions are explained as follows:

4.1 Mounting Methods of Pin Insertion Type Package

Insert lead pins into the PCG through-holes (usually about ¼0.8mm). Soak leads in a wave solder tub.

Lead pins held by the through-holes enable handling of the package through the soldering process, and facilitate automated soldering. When soldering leads in the wave solder tub, do not get solder on the package.

4.2 Mounting Method of Surface Mount Type Package

Apply the specified quantity of solder paste to the pattern on any printed board by the screen printing method, to temporarily fix the package to the board. The solder paste melts when heated in a reflowing furnace, and package leads and the pattern of the printed board are fixed by the surface tension of the melted solder and self alignment.

The size of the pattern where leads are attached should be 1.1 to 1.3 times the leads' width, depending on paste material or furnace adjustment.

The temperature of the reflowing furnace is dependent on packaging material and type. Fig. 2 lists the adjustment of the reflowing furnace for FPP. Pre-heat the furnace to 150° C. Surface temperature of the resin should be kept at 235° C maximum for 10 minutes or less.

(1) The temperature of the leads should be kept at 260°C for 10 minutes or less.
(2) The temperature of the resin should be kept at 235°C for 10 minutes or less.
(3) Below is shown the temperature profile when soldering a package by the reflowing method.

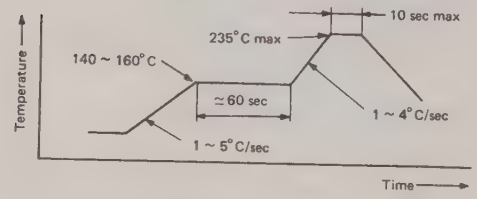


Figure 2 Reflowing Furnace Adjustment for FPP

Employ adequate heating or temperature control equipment to prevent damage to the plastic package epoxy-resin material. When using an infrared heater, avoid long exposure at temperatures higher than the glass transition point of epoxy-resin (about 150° C), which may cause package damage and loss of reliability characteristics. Equalize the temperature inside and outside of packages by reducing the heat of the upper surface of the packages.

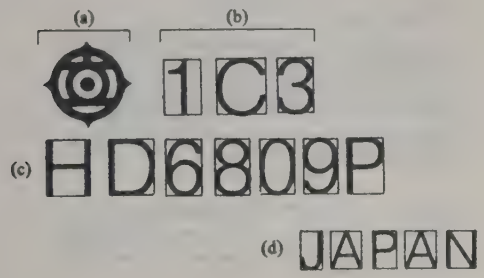
FPP leads may easily bend in shipment or during handling, and impact soldering onto the printed board. Heat the bent leads again with a soldering iron to reshape them.

Use a rosin flux when soldering. Do not use chloric flux because the chlorine in the flux has a tendency to remain on the leads and reduce reliability. Use alcohol, chlorothene or freon to wash away rosin flux from packages. These solvents should not remain on the packages for an excessive length of time, because the package markings may disappear.

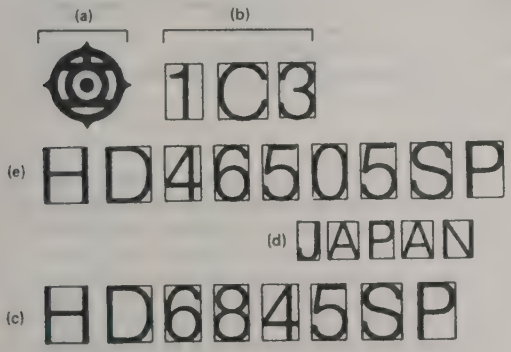
5. Marking

The Hitachi trademark, product type No., and other markings are printed on packages as shown in the following examples. Case I and Case II are examples of markings and Nos. Case I applies to products which have only a standard type No., while Case II applies to products which have an old type No. and a standard type No.

Case I; Includes a standard type No.



Case II; Includes an old type No. and a standard type No.



Meaning of Each Mark

(a)	Hitachi Trademark
(b)	Lot Code
(c)	Standard Type No.
(d)	Japan Mark
(e)	Old Type No.

QUALITY ASSURANCE

1. VIEWS ON QUALITY AND RELIABILITY

Basic views on quality at Hitachi are to meet the individual uers' required quality level and maintain a general quality level equal to or above that of the general market. The quality required by the user may be specified by contract, or may be indefinite. In either case, efforts are made to assure reliable performance in actual operating circumstances. Quality control during the manufacturing process, and quality awareness from design through production lead to product quality and customer satisfaction. Our quality assurance technique consists basically of the following steps:

(1) Build in reliability at the design stage of new product development.
(2) Build in quality at all steps in the manufacturing process.
(3) Execute stringent inspection and reliability confirmation of final products.
(4) Enhance quality levels through field data feed back.
(5) Cooperate with research laboratories for higher quality and reliability.

With the views and methods mentioned above, utmost efforts are made to meet users' requirements.

2. RELIABILITY DESIGN OF SEMICONDUCTOR DEVICES

2.1 Reliability Targets

The reliability target is an important factor in sales, manufacturing, performance, and price. It is not adequate to set a reliability target based on a single set of common test conditions. The reliability target is set based on many factors:

(1) End use of semiconductor device.
(2) End use of equipment in which device is used.
(3) Device manufacturing process.
(4) End user manufacturing techniques.
(5) Quality control and screening test methods.
(6) Reliability target of system.

2.2 Reliability Design

The following steps are taken to meet the reliability targets:

(1) Design Standardization
As for design rules, critical items pertaining to quality and reliability are always studied at circuit design, device design, layout design, etc. Therefore, as long as standardized processing and materials are used the reliability risk is extremely small even in the case of new development devices, with the exception of special requirements imposed by functional needs.

(2) Device Design
It is important for the device design to consider total balance of process, structure, circuit, and layout design, especially in the case where new processes and/or new materials are employed. Rigorous technical studies are conducted prior to device development.

(3) Reliability Evaluation by Functional Test
Functional Testing is a useful method for design and process reliability evaluation of IC's and LSI devices which have complicated functions.

The objectives of Functional Test are:

- Determining the fundamental failure mode.
- Analysis of relation between failure mode and manufacturing process.
- Analysis of failure mechanism.
- Establishment of QC points in manufacturing process.

2.3 Design Review

Design Review is an organized method to confirm that a design satisfies the performance required and meets design specifications. In addition, design review helps to insure quality and reliability of the finished products. At Hitachi, design review is performed from the planning stage to production for new products, and also for design changes on existing products. Items discussed and considered at design review are:

(1) Description of the products based on design documents.
(2) From the standpoint of each participant, design documents are studied, and for points needing clarification, further investigation will be carried out.
(3) Specify quality control and test methods based on design documents and drawings.
(4) Check process and ability of manufacturing line to achieve design goal.
(5) Preparation for production.
(6) Planning and execution of sub-programs for design changes proposed by individual specialists,

for test, experiments, and calculations to confirm the design changes.

(7) Analysis of past failures with similar devices, discussion of methods to prevent them, and planning and execution of test programs to confirm success.

3. QUALITY ASSURANCE SYSTEM

3.1 Activity of Quality Assurance

General views of overall quality assurance in Hitachi are as follows:

(1) Problems in each individual process should be solved in the process. Therefore, at the finished product stage the potential failure factors have been removed.

(2) Feedback of information is used to insure a satisfactory level of ability process.

3.2 Quality Approval

To insure quality and reliability, quality approval is carried out at the preproduction stage of device design, as described in section 2. Our views on quality approval are:

(1) A third party executes approval objectively from the standpoint of the customer.

(2) Full consideration is given to past failures and information from the field.

(3) No design change or process change without QA approval.

(4) Parts, materials, and processes are closely monitored.

(5) Control points are established in mass production after studying the process abilities and variables.

3.3 Quality and Reliability Control at Mass Production

Quality control is accomplished through division of functions in manufacturing, quality assurance, and other related departments. The total function flow is shown in Fig. 2. The main points are described below.

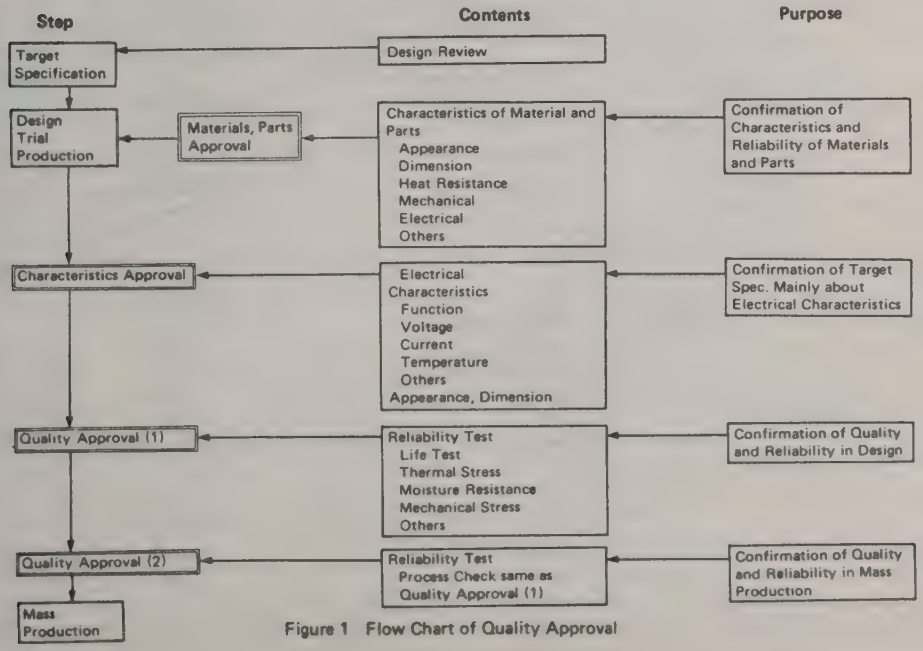


Figure 1 Flow Chart of Quality Approval

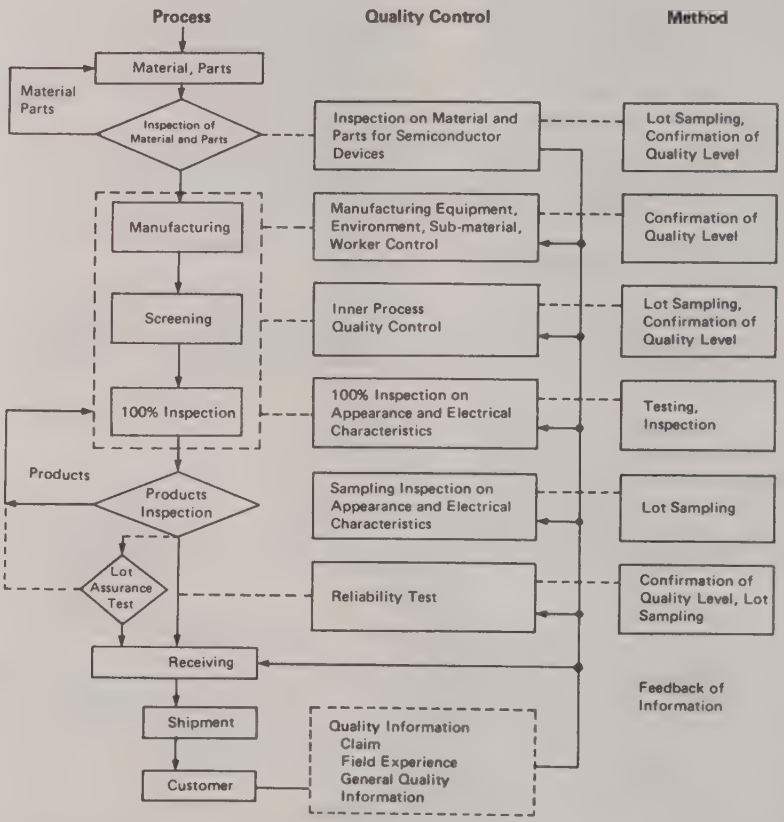


Figure 2 Flow Chart of Quality Control in Manufacturing Process

3.3.1 Quality Control of Parts and Materials

As semiconductor devices tend towards higher performance and higher reliability, the importance of quality control of parts and materials becomes paramount. Items such as crystals, lead frames, fine wire for wire bonding, packages, and materials needed in manufacturing processes such as masks and chemicals, are all subject to rigorous inspection and control. Incoming inspection is performed based on the purchase specification and drawing. The sampling is executed based mainly on MIL-STD-105D.

The other activities of quality assurance are as follows:

(1) Outside vendor technical information meeting.
(2) Approval and guidance of outside vendors.
(3) Chemical analysis and test.

The typical check points of parts and materials are shown in Table 1.

Table 1 Quality Control Check Points of Material and Parts (Example)

Material, Parts	Important Control Items	Point for Check
Wafer	Appearance	Damage and Contamination on Surface
	Dimension	Flatness
	Sheet Resistance	Resistance
	Defect Density	Defect Numbers
	Crystal Axis	
Mask	Appearance	Defect Numbers, Scratch
	Dimension	Dimension Level
	Resistoration	
	Gradation	Uniformity of Gradation
Fine Wire for Wire Bonding	Appearance	Contamination, Scratch, Bend, Twist
	Dimension	
	Purity	Purity Level
	Elongation Ratio	Mechanical Strength
Frame	Appearance	Contamination, Scratch
	Dimension	Dimension Level
	Processing Accuracy	
	Plating	Bondability, Solderability
	Mounting Characteristics	Heat Resistance
Ceramic Package	Appearance	Contamination, Scratch
	Dimension	Dimension Level
	Leak Resistance	Airtightness
	Plating	Bondability, Solderability
	Mounting Characteristics	Heat Resistance
	Electrical Characteristics	
	Mechanical Strength	Mechanical Strength
Plastic	Composition	Characteristics of Plastic Material
	Electrical Characteristics	
	Thermal Characteristics	
	Molding Performance	Molding Performance
	Mounting Characteristics	Mounting Characteristics

3.3.2 Inner Process Quality Control

Inner Process Quality Control performs very important functions in quality assurance of semiconductor devices. The manufacturing Inner Process Quality Control is shown in Fig. 3.

(1) Quality Control of Semi-final Products and Final Products

Potential failure factors of semiconductor devices are removed in the manufacturing process. To achieve this, check points are set-up in each process and products which have potential failure factors are not moved to the next process step. Manufacturing lines are rigidly selected and tight inner process quality controls are executed—rigid checks in each process and each lot, 100% inspection to remove failure factors caused by manufacturing variables and high temperature aging and temperature cycling. Elements of inner process quality control are as follows:

- Condition control of equipment and workers environment and random sampling of semi-final products.
- Suggestion system for improvement of work.
- Education of workers.
- Maintenance and improvement of yield.
- Determining quality problems, and implementing countermeasures.
- Transfer of quality information.

(2) Quality Control of Manufacturing Facilities and Measuring Equipment

Manufacturing equipment is improving as higher performance devices are needed. At Hitachi, the automation of manufacturing equipment is encouraged. Maintenance Systems maintain operation of high performance equipment. There are daily inspections which are performed based on related specifications. Inspection points are listed in the specification and are checked one by one to prevent any omission. As for adjustment and maintenance of measuring equipment, specifications are checked one by one to maintain and improve quality.

(3) Quality Control of Manufacturing Circumstances and Sub-Materials

The quality and reliability of semiconductor devices are highly affected by the manufacturing process. Therefore, controls of manufacturing circum-

stances such as temperature, humidity and dust, and the control of submaterials, like gas, and pure water used in a manufacturing process, are intensively executed.

Dust control is essential to realize higher integration and higher reliability of devices. At Hitachi, maintenance and improvement of cleanliness at manufacturing sites is accomplished through attention to buildings, facilities, air conditioning systems, delivered materials, clothes, work environment, and periodic inspection of floating dust concentration.

3.3.3 Final Product Inspection and Reliability Assurance

(1) Final Product Inspection

Lot inspection is done by the quality assurance

Process	Control Point		Purpose of Control
▽ Purchase of Material			
Wafer	Wafer	Characteristics, Appearance	Scratch, Removal of Crystal Defect Wafer
Surface Oxidation	Oxidation		Assurance of Resistance
Inspection on Surface Oxidation		Appearance, Thickness of Oxide Film	Pinhole, Scratch
Photo Resist	Photo Resist		
Inspection on Photo Resist		Dimension, Appearance	Dimension Level
◇ PQC Level Check			Check of Photo Resist
Diffusion	Diffusion	Diffusion Depth, Sheet Resistance	Diffusion Status
Inspection on Diffusion		Gate Width	Control of Basic Parameters (V_{TH}, etc.) Cleanness of surface, Prior Check of V_{IH} Breakdown Voltage Check
◇ PQC Level Check		Characteristics of Oxide Film Breakdown Voltage	
Evaporation	Evaporation	Thickness of Vapor Film, Scratch, Contamination	Assurance of Standard Thickness
Inspection on Evaporation			
◇ PQC Level Check			
Wafer Inspection	Wafer	Thickness, V_{TH} Characteristics	Prevention of Crack, Quality Assurance of Scribe
Inspection on Chip Electrical Characteristics	Chip	Electrical Characteristics	
Chip Scribe		Appearance of Chip	
Inspection on Chip Appearance			
◇ PQC Lot Judgement			
Frame			
Assembling	Assembling	Appearance after Chip Bonding	Quality Check of Chip Bonding
		Appearance after Wire Bonding	Quality Check of Wire Bonding
◇ PQC Level Check		Pull Strength, Compression Width, Shear Strength	Prevention of Open and Short
Inspection after Assembling		Appearance after Assembling	
◇ PQC Lot Judgement			
Package			
Sealing	Sealing	Appearance after Sealing Outline, Dimension	Guarantee of Appearance and Dimension
◇ PQC Level Check	Marking	Marking Strength	
Final Electrical Inspection			
◇ Failure Analysis		Analysis of Failures, Failure Mode, Mechanism	Feedback of Analysis Information
Appearance Inspection			
Sampling Inspection on Products			
Receiving			
Shipment			

Figure 3 Example of Inner Process Quality Control

department for products which were judged good in 100% test . . . the final process in manufacturing. Though 100% yield is expected, sampling inspection is executed to prevent mixture of bad product by mistake. The inspection is executed not only to confirm that the products have met the users' requirements but also to consider potential quality factors. Lot inspection is executed based on MIL-STD-105D.

(2) Reliability Assurance Tests
To assure the reliability of semiconductor devices, reliability tests and tests on individual manufacturing lots that are required by the user, are periodically performed.

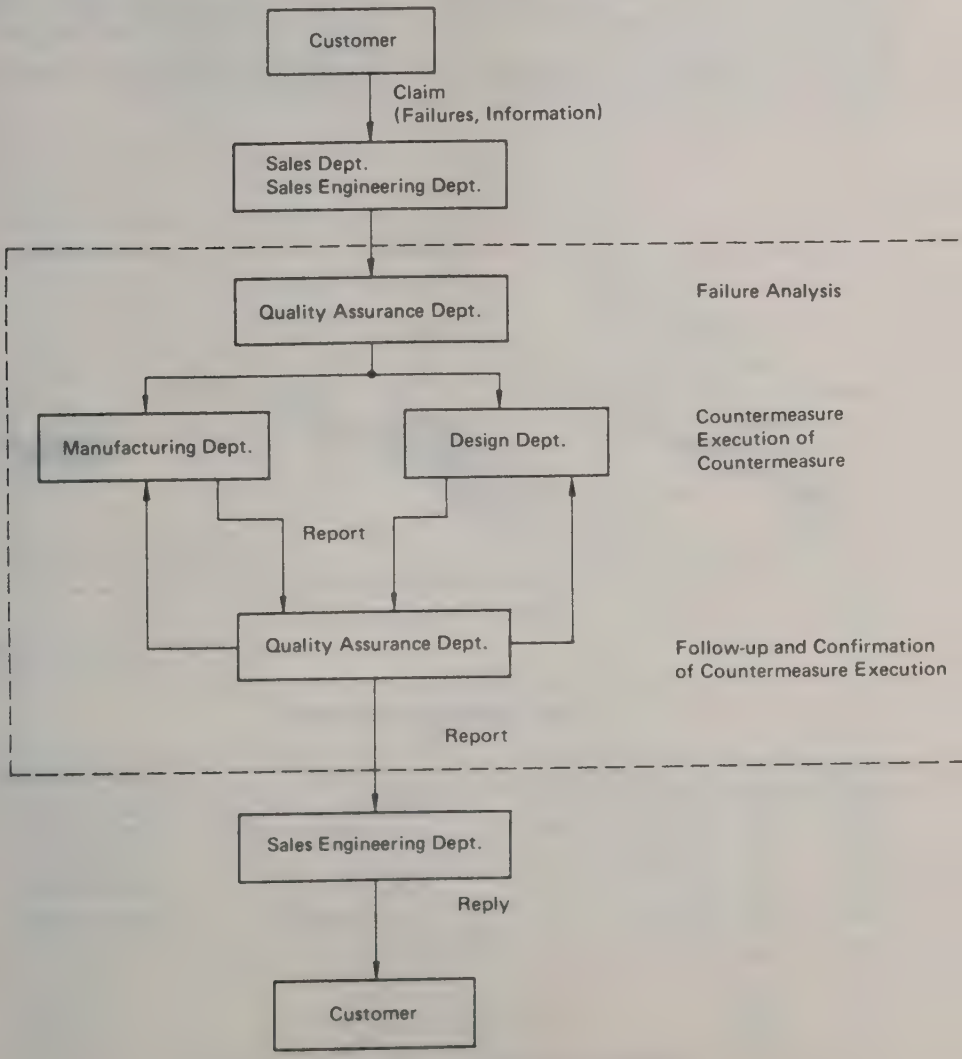


Figure 4 Process Flow Chart of Field Failure

RELIABILITY TEST DATA

1. INTRODUCTION

Microcomputers provide high reliability and quality to meet the demands of increased function, enlarging scale, and widening application. Hitachi has improved the quality level of microcomputer products by evaluating reliability, building quality into the manufacturing process, strengthening inspection techniques, and analyzing field data.

The following reliability and quality assurance data for Hitachi 8-bit and 16-bit multi-chip microcomputers indicates results from test and failure analysis.

2. PACKAGE AND CHIP STRUCTURE

2.1 Packaging

Production output and application of plastic packaging continues to increase, expanding to automobile measuring and control systems, and computer terminal equipment operating under severe conditions. To meet this demand, Hitachi has significantly improved moisture resistance and operational stability in the plastic manufacturing process.

Plastic and side-brazed ceramic package structures are shown in Figure 1 and Table 1.

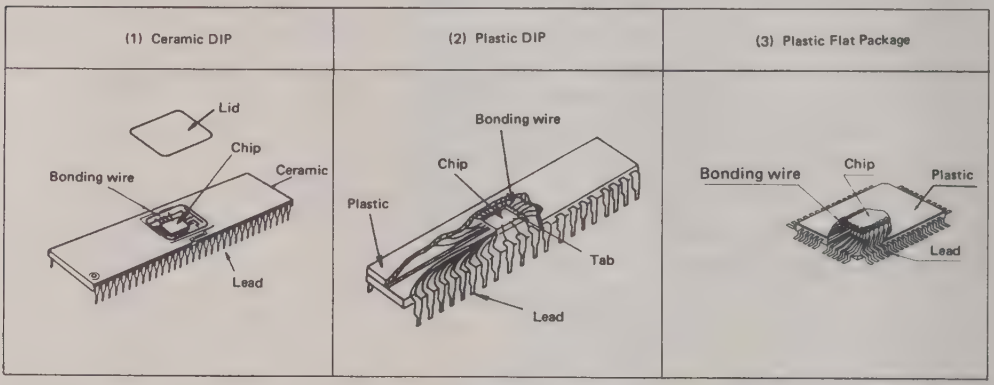


Figure 1 Package Structure

Table 1 Package Material and Properties

Item	Ceramic DIP	Plastic DIP	Plastic Flat Package
Package	Alumina	Epoxy	Epoxy
Lead	Tin plating Brazed Alloy 42	Solder dipping Alloy 42 or Cu	Solder plating Alloy 42
Seal	Au-Sn Alloy	N.A	N.A
Die bond	Au-Si	Au-Si or Ag paste	Au-Si or Ag paste
Wire bond	Ultrasonic	Thermo compression	Thermo compression
Wire	Al	Au	Au

2.2 Chip Structure

Hitachi microcomputers are produced in NMOS E D technology or low power CMOS technology. Si-gate process is used in both types to achieve high reliability and density.

Chip structure and basic circuit are shown in Figure 2.

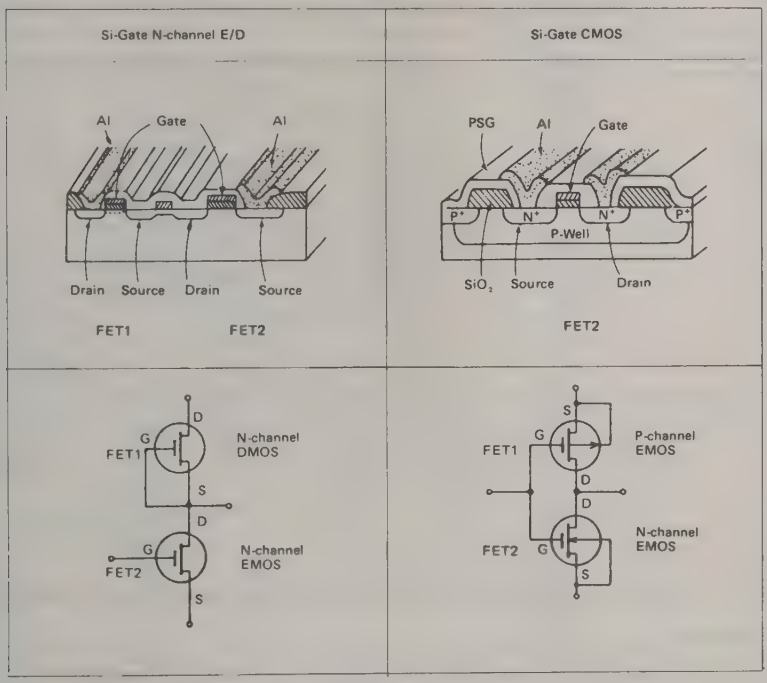


Figure 2 Chip Structure and Basic Circuit

3. QUALITY QUALIFICATION AND EVALUATION

3.1 Reliability Test Methods

Reliability test methods shown in Table 2 are used to qualify and evaluate new products and processes.

Table 2 Reliability Test Methods

Test Items	Test Condition	MIL-STD-883B Method No.
Operating Life Test	125°C, 1000hr	1005,2
High Temp, Storage	Tstg max, 1000hr	1008,1
Low Temp, Storage	Tstg min, 1000hr	
Steady State Humidity	65°C 95%RH, 1000hr	
Steady State Humidity Biased	85°C 85%RH, 1000hr	
Temperature Cycling	-55°C ~ 150°C, 10 cycles	1010,4
Temperature Cycling	-20°C ~ 125°C, 200 cycles	
Thermal Shock	0°C ~ 100°C, 100 cycles	1011,3
Soldering Heat	260°C, 10 sec	
Mechanical Shock	1500G 0.5 msec, 3 times/X, Y, Z	2002,2
Vibration Fatigue	60Hz 20G, 32hrs/X, Y, Z	2005,1
Variable Frequency	20~2000Hz 20G, 4 min/X, Y, Z	2007,1
Constant Acceleration	20000G, 1 min/X, Y, Z	2001,2
Lead Integrity	225gr, 90° 3 times	2004,3

3.2 Reliability Test Result

Reliability Test Results of 8-bit multi-chip microcomputer devices is shown in Table 3 to Table 7, and of 16-bit microprocessor HD68000 in Table 8, Table 9.

Table 3 Dynamic Life Test (8-bit multi-chip microcomputer)

Device Type	Sample Size	Component Hours	Failures
HD6800	248 pcs	248000	0
HD6802	452	153712	1*
HD6809	85	85000	0
HD6821	399	266368	1*
HD6850	158	158000	0
HD6852	170	125816	0
HD6846	69	69000	0
HD6843	66	66000	0
HD6844	80	69000	0
HD6845S	88	55000	0
HD6840	64	64000	0
HD46508	140	140000	0
HD146818	44	44000	0
HD6350	70	70000	0
HD6321	45	45000	0
Total	2178	1658896	2

*leakage current

Estimated Field Failure Rate
= 0.01% / 1000 hrs at Ta = 75°C
(Activation Energy = 0.7eV, Confidence Level 60%)

Table 4 High Temperature, High Humidity Test (8-bit multi-chip microcomputer) (Moisture Resistance Test)

(1) 85°C 85%RH Bias Test

Device Type	Vcc Bias	168 hrs	500 hrs	1000 hrs
HD6800P	5.5V	0/45	0/45	0/45
HD6802P	5.5V	0/38	0/38	0/38
HD6809P	5.5V	0/22	0/22	0/22
HD6850P	5.5V	0/45	0/45	0/45
HD6852P	5.5V	0/22	0/22	0/22
HD6843P	5.5V	0/22	0/22	0/22
HD6844P	5.5V	0/22	0/22	0/22
HD6845SP	5.5V	0/137	0/137	0/137
HD6840P	5.5V	0/22	0/22	0/22
HD46508P	5.5V	0/22	0/22	0/22
HD146818P	5.5V	0/22	0/22	0/22
HD6321P	5.5V	0/22	0/22	0/22
Total		0/441	0/441	0/441

(2) High Temperature-High Humidity Storage Life Test

Device Type	Condition	168 hrs	500 hrs	1000 hrs
HD6800P	65°C 95%RH	0/22	0/22	0/22
HD6802P	80°C 90%RH	0/22	0/22	0/22
HD6802P	65°C 95%RH	0/38	0/38	0/38
HD6809P	65°C 95%RH	0/45	0/45	0/45
HD6850P	65°C 95%RH	0/135	0/135	0/135
HD6850P	80°C 90%RH	0/22	0/22	0/22
HD6852P	85°C 95%RH	0/22	0/22	0/22
HD6843P	80°C 95%RH	0/22	0/22	0/22
HD6844P	80°C 90%RH	0/22	0/22	0/22
HD6845SP	80°C 90%RH	0/22	0/22	0/22
HD6840P	65°C 95%RH	0/22	0/22	0/22
HD46508P	65°C 95%RH	0/70	0/70	0/70
HD146818P	65°C 95%RH	0/45	0/45	0/45
HD6350P	65°C 95%RH	0/70	0/70	0/70
HD6321P	65°C 95%RH	0/70	0/70	0/70

(3) Pressure Cooker Test
(Condition ; 2atm 121°C)

Device Type	40 hrs	60 hrs	100 hrs	200 hrs
HD6800P	0/42	0/42	0/42	–
HD6802P	0/22	0/22	0/22	–
HD6821P	0/44	0/44	0/44	–
HD6850P	0/22	0/22	0/22	–
HD6843P	0/22	0/22	0/22	–
HD6845SP	0/43	0/43	0/43	1*/43
HD6846P	0/13	0/13	0/13	–
HD46508P	0/45	0/45	0/45	–
HD146818P	0/22	0/22	0/22	–
HD6350P	0/22	0/22	0/22	–
HD6321P	0/22	0/22	0/22	–

*Aluminum corrosion

(4) MIL-STD-883B Moisture Resistance Test
(Condition; 65°C ~ −10°C, over 90%RH, Vcc = 5.5V)

Device Type	10 cycles	20 cycles	40 cycles
HD6800P	0/25	0/25	0/25
HD6802P	0/25	0/25	0/25
HD6821P	0/25	0/25	0/25
HD6846P	0/25	0/25	0/25

Table 5 Temperature Cycling Test (8-bit multi-chip microcomputer) (-55°C ~ 25°C ~ 150°C)

Device Type	10 cycles	100 cycles	200 cycles
HD6800P	0/453	0/44	0/44
HD6802P	0/502	0/77	0/77
HD6809P	0/202	0/45	0/45
HD6821P	0/420	0/44	1*/44
HD6850P	0/151	0/38	0/38
HD6852P	0/149	0/38	0/38
HD6843P	0/247	0/44	0/44
HD6844P	0/150	0/44	0/44
HD6845SP	0/358	0/76	0/76
HD6846P	0/150	0/22	0/22
HD6840P	0/148	0/22	0/22
HD46508P	0/207	0/44	0/44
HD146818P	0/103	0/22	0/22
HD6350P	0/150	0/45	0/45
HD6321P	0/120	0/45	0/45

* Static damage

Table 6 High Temperature, Low Temperature Storage Life Test (8-bit multi-chip microcomputer)

Device	Temperature	168 hrs	500 hrs	1000 hrs
MPU total	150°C	0/88	0/88	0/88
	-55°C	0/76	0/76	0/76
Peripheral total	150°C	0/110	0/110	0/110
	-55°C	0/88	0/88	0/88

Table 7 Mechanical and Environmental Test (8-bit multi-chip microcomputer)

Test Item	Condition	Plastic DIP		Flat Plastic Package	
		Sample Size	Failure	Sample Size	Failure
Thermal Shock	0°C ~ 100°C 10 cycles	110	0	100	0
Soldering Heat	260°C, 10 sec.	164	0	20	0
Salt Water Spray	35°C, NaCl 5% 24 hrs	110	0	20	0
Solderability	230°C, 5 sec. Rosin flux	159	0	34	0
Drop Test	75cm, maple board 3 times	110	0	20	0
Mechanical Shock	1500G, 0.5 ms 3 times/X, Y, Z	110	0	20	0
Vibration Fatigue	60 Hz, 20G 32 hrs/X, Y, Z	110	0	20	0
Vibration Variable Freq.	100 ~ 2000 Hz 20G, 4 times/X, Y, Z	110	0	20	0
Lead Integrity	225 g, 90° Bonding 3 times	110	0	20	0

Table 8 Dynamic Life Test (16-bit multi-chip microcomputer)

Device Type	Condition		168 hrs	500 hrs	1000 hrs
	Ta	Vcc			
HD68000	125°C	5.5V	0/62	0/62	0/62
	150°C	5.5V	0/52	0/52	0/52
HD68450	125°C	5.5V	0/44	0/44	0/44
	150°C	5.5V	0/32	0/32	0/32

Estimated Field Failure Rate
= 0.013%/1000 hrs at Ta = 75°C
(Activation Energy 0.7eV, Confidence Level 60%)

Table 9 Mechanical and Environmental Test (16-bit multi-chip microcomputer)

Test Item	Condition	Device Type	
		HD68000	HD68450
High Temperature Storage	Ta = 295°C, 1000 hrs	0/42	0/22
Low Temperature Storage	Ta = −55°C, 1000 hrs	0/42	0/42
Temperature Cycling (1)	−55°C ~ 25°C ~ 150°C 10 cycles	0/189	0/45
Temperature Cycling (2)	−20°C ~ 25°C ~ 125°C 500 cycles	0/44	0/22
Thermal Shock	−55°C ~ 125°C 15 cycles	0/44	0/22
Soldering heat	260°C, 10 sec	0/44	0/22
Solderability	230°C, 5 sec	0/44	0/22
Mechanical Shock	1500G, 0.5 msec 3 times/X, Y, Z	0/44	0/22
Vibration Variable Freq.	20 ~ 2000 Hz, 20G 3 times/X, Y, Z	0/44	0/22
Constant Acceleration	20000G 1 min/X, Y, Z	0/44	0/22

4. PRECAUTIONS

4.1 Storage

To prevent deterioration of electrical characteristics, solderability, appearance or structure, Hitachi recommends semiconductor devices be stored as follows:

(1) Store in ambient temperatures of 5 to 30° C, with a relative humidity of 40 to 60%.
(2) Store in a clean, dust- and active gas-free environment.
(3) Store in conductive containers to prevent static electricity.
(4) Store without any physical load.
(5) When storing devices for an extended period, store in an unfabricated form, to minimize corrosion of pre-formed lead wires.
(6) Unsealed chips should be stored in a cool, dry, dark and dust-free environment. Assembly should be performed within 5 days of unpacking. Devices can be stored for up to 20 days in dry nitrogen gas with a dew point at -30° C or less.
(7) Prevent condensation during storage due to rapid temperature changes.

4.2 Transportation

General precautions for electronic components are applicable in transporting semiconductors, units incorporating semiconductors, and other similar systems. In addition, Hitachi recommends the following:

(1) When transporting semiconductor devices or printed circuit boards, minimize mechanical vibration and shock. Use containers or jigs which will not induce static electricity as a result of vibration. Use of an electrically conductive container or aluminum foil is recommended.
(2) To prevent device deterioration from clothing-induced static electricity, workers should be properly grounded while handling devices. Use of a 1 M ohm resistor is recommended to prevent electric shock.
(3) When transporting printed curcuit boards containing semiconductor devices, suitable preventive measures against static electricity must be taken. Voltage build-up can be avoided by shorting the card-edge terminals. When a belt conveyor is used, apply some surface treatment to prevent build-up of electrical charge.
(4) Minimize mechanical vibration and shock when transporting semiconductor devices or printed circuit boards.

4.3 Handling for Measurement

Avoid static electricity, noise and voltage surge when measuring or mounting devices. Precaution should be taken against current leakage through terminals and housings of curve tracers, synchroscopes, pulse generators, and DC power sources.

When testing devices, prevent voltage surges from the tester, attached clamping circuit, and any excessive voltage possible through accidental contact.

In inspecting a printed circuit board, power should not be applied if any solder bridges or foreign matter is present.

4.4 Soldering

Semiconductor devices should not be exposed to high temperatures for excessive periods. Soldering must be performed consistent with temperature conditions of 260° C for 10 seconds, 350° C for 3 seconds, and at a distance of 1 to 1.5mm from the end of the device package.

A soldering iron with secondary voltage supplied through a grounded transformer is recommended to protect against leakage current. Use of alkali or acid flux, which may corrode the leads, is not recommended.

4.5 Removing Residual Flux

Detergent or ultrasonic removal of residual flux from circuit boards is necessary to ensure system reliability. Selection of detergent type and cleaning conditions are important factors.

When chloric detergent is used for plastic packaged devices, care must be taken against package corrosion. Extended cleaning periods and excessive temperature conditions can cause the chip coating to swell due to solvent permeation. Hitachi recommends use of Lotus and Dyfron solvents. Trichloroethylene solvent is not suitable.

The following conditions are advisable for ultrasonic cleaning:

- Frequency: 28 to 29 k Hz (to avoid device resonation)
- Ultrasonic output: 15W/1
- Keep devices from making direct contact with power generator
- Cleaning time: Less than 30 seconds.

DATA SHEETS

8-BIT MULTI-CHIP MICROCOMPUTER

Preliminary data sheets herein contain information on new products. Specifications and information are subject to change without notice.

Advance Information data sheets herein contain information on a product under development. Hitachi reserves the right to change or discontinue these products without notice.

HD6803, HD6803-1

MPU (Micro Processing Unit)

The HD6803 MPU is an 8-bit micro processing unit which is compatible with the HMCS6800 family of parts. The HD6803 MPU is object code compatible with the HD6800 with improved execution times of key instructions plus several new 16-bit and 8-bit instruction including an 8 × 8 unsigned multiply with 16-bit result. The HD6803 MPU can be expanded to 65k bytes. The HD6803 MPU is TTL compatible and requires one +0.5 volt power supply. The HD6803 MPU has 128 bytes of RAM, Serial Communications Interface (S.C.I.), and parallel I/O as well as a three function 16-bit timer. Features and Block Diagram of the HD6803 include the following:

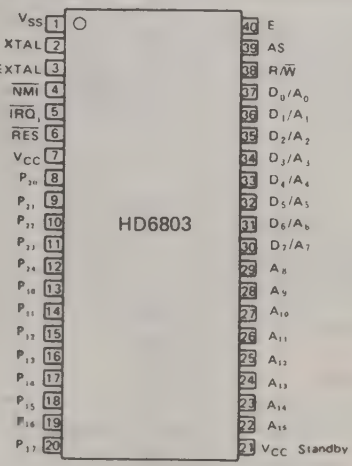
HD6803P
HD6803P-1

(DP-40)

■ FEATURES

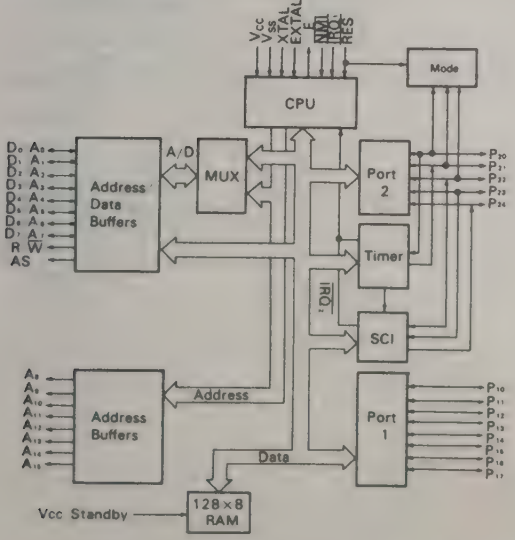
- Expanded HMCS6800 Instruction Set
- 8 x 8 Multiply
- On-Chip Serial Communications Interface (S.C.I.)
- Object Code Compatible with The HD6800 MPU
- 16-Bit Timer
- Expandable to 65k Bytes
- Multiplexed Address and Data
- 128 Bytes of RAM (64 Bytes Retainable On Power Down)
- 13 Parallel I/O Lines
- Internal Clock/Divided-By-Four
- TTL Compatible Inputs and Outputs
- Interrupt Capability
- Compatible with MC6803 and MC6803-1

■ PIN ARRANGEMENT



(Top View)

■ BLOCK DIAGRAM



■ TYPE OF PRODUCTS

Type No.	Bus Timing
HD6803	1.0MHz
HD6803-1	1.25MHz

■ ABSOLUTE MAXIMUM RATINGS

Item	Symbol	Value	Unit
Supply Voltage	V_{CC} *	-0.3 ~ +7.0	V
Input Voltage	V_{in} *	-0.3 ~ +7.0	V
Operating Temperature	T_{opr}	0 ~ +70	°C
Storage Temperature	T_{stg}	- 55 ~ +150	°C

* With respect to V_{SS} (SYSTEM GND)

[NOTE] Permanent LSI damage may occur if maximum ratings are exceeded. Normal operation should be under recommended operating conditions. If these conditions are exceeded, it could affect reliability of LSI.

■ ELECTRICAL CHARACTERISTICS

● DC CHARACTERISTICS (V_{CC} =5.0V±5%, V_{SS} = 0V, Ta = 0~+70°C, unless otherwise noted.)

Item		Symbol	Test Condition	min	typ	max	Unit
Input "High" Voltage	$\overline{RES}$	V_{IH}		4.0	–	V_{CC}	V
	Other Inputs*			2.0	–	V_{CC}	
Input "Low" Voltage	All Inputs*	V_{IL}		-0.3	–	0.8	V
Input Load Current	EXTAL	$\|I_{in}\|$	V_{in} = 0 ~ V_{CC}	–	–	0.8	mA
Input Leakage Current	$\overline{NMI}$, $\overline{IRQ_1}$, $\overline{RES}$	$\|I_{in}\|$	V_{in} = 0 ~ 5.25V	–	–	2.5	μA
Three State (Offset) Leakage Current	P_{10} ~ P_{17}, D_0/A_0 ~ D_7/A_7	$\|I_{TSI}\|$	V_{in} = 0.5 ~ 2.4V	–	–	10	μA
	P_{20} ~ P_{24}			–	–	100	
Output "High" Voltage	D_0/A_0 ~ D_7/A_7	V_{OH}	I_{LOAD} = -205 μA	2.4	–	–	V
	A_8 ~ A_{15}, E, R/$\overline{W}$, AS		I_{LOAD} = -145 μA	2.4	–	–	
	Other Outputs		I_{LOAD} = -100 μA	2.4	–	–	
Output "Low" Voltage	All Outputs	V_{OL}	I_{LOAD} = 1.6 mA	–	–	0.5	V
Darlington Drive Current	P_{10} ~ P_{17}	$-I_{OH}$	V_{out} = 1.5V	1.0	–	10.0	mA
Power Dissipation		P_D		–	–	1200	mW
Input Capacitance	D_0/A_0 ~ D_7/A_7	C_{in}	V_{in} = 0V, Ta = 25°C, f = 1.0 MHz	–	–	12.5	pF
	Other Inputs			–	–	10.0	
V_{CC} Standby	Powerdown	V_{SBB}		4.0	–	5.25	V
	Operating	V_{SB}		4.75	–	5.25	
Standby Current	Powerdown	I_{SBB}	V_{SBB} = 4.0V	–	–	8.0	mA

*Except Mode Programming Levels.

• AC CHARACTERISTICS

BUS TIMING (V_{CC} = 5.0V ± 5%, V_{SS} = 0V, Ta = 0 ~ +70°C, unless otherwise noted.)

Item		Symbol	Test Condition	HD6803			HD6803-1			Unit
				min	typ	max	min	typ	max	
Cycle Time		t_{cyc}	Fig. 1	1	–	10	0.8	–	10	μs
Address Strobe Pulse Width "High" *		PW_{ASH}		200	–	–	150	–	–	ns
Address Strobe Rise Time		t_{ASr}		5	–	50	5	–	50	ns
Address Strobe Fall Time		t_{ASf}		5	–	50	5	–	50	ns
Address Strobe Delay Time *		t_{ASD}		60	–	–	30	–	–	ns
Enable Rise Time		t_{Er}		5	–	50	5	–	50	ns
Enable Fall Time		t_{Ef}		5	–	50	5	–	50	ns
Enable Pulse Width "High" Time *		PW_{EH}		450	–	–	340	–	–	ns
Enable Pulse Width "Low" Time *		PW_{EL}		450	–	–	350	–	–	ns
Address Strobe to Enable Delay Time *		t_{ASED}		60	–	–	30	–	–	ns
Address Delay Time		t_{AD}		–	–	260	–	–	260	ns
Address Delay Time for Latch *		t_{ADL}		–	–	270	–	–	260	ns
Data Set-up Write Time		t_{DSW}		225	–	–	115	–	–	ns
Data Set-up Read Time		t_{DSR}		80	–	–	70	–	–	ns
Data Hold Time	Read	t_{HR}		10	–	–	10	–	–	ns
	Write	t_{HW}		20	–	–	20	–	–	
Address Set-up Time for Latch *		t_{ASL}		60	–	–	50	–	–	ns
Address Hold Time for Latch		t_{AHL}		20	–	–	20	–	–	ns
Address Hold Time		t_{AH}		20	–	–	20	–	–	ns
Peripheral Read Access Time (Multiplexed Bus)*		(t_{ACCM})		–	–	(600)	–	–	(420)	ns
Oscillator stabilization Time		t_{RC}	Fig. 8	100	–	–	100	–	–	ms
Processor Control Set-up Time		t_{PCS}	Fig. 7,8	200	–	–	200	–	–	ns

*These timings change in approximate proportion to t_{cyc}. The figures in this characteristics represent those when t_{cyc} is minimum (= in the highest speed operation).

PERIPHERAL PORT TIMING (V_{CC} = 5.0V ± 5%, V_{SS} = 0V, Ta = 0 ~ +70°C, unless otherwise noted.)

Item		Symbol	Test Condition	min	typ	max	Unit
Peripheral Data Setup Time	Port 1, 2	t_{PDSU}	Fig. 2	200	–	–	ns
Peripheral Data Hold Time	Port 1, 2	t_{PDH}	Fig. 2	200	–	–	ns
Delay Time, Enable Negative Transition to Peripheral Data Valid	Port 1, 2*	t_{PWD}	Fig. 3	–	–	400	ns

* Except P_{21}

TIMER, SCI TIMING (V_{CC} = 5.0V ±5%, V_{SS} = 0V, Ta = 0 ~ +70°C, unless otherwise noted.)

Item	Symbol	Test Condition	min	typ	max	Unit
Timer Input Pulse Width	t_{PWT}		$2t_{cyc}+200$	–	–	ns
Delay Time, Enable Positive Transition to Timer Out	t_{TOD}	Fig. 4	–	–	600	ns
SCI Input Clock Cycle	t_{Scyc}		1	–	–	t_{cyc}
SCI Input Clock Pulse Width	t_{PWSCK}		0.4	–	0.6	t_{Scyc}

MODE PROGRAMMING (V_{CC} = 5.0V ±5%, V_{SS} = 0V, Ta = 0 ~ +70°C, unless otherwise noted.)

Item		Symbol	Test Condition	min	typ	max	Unit
Mode Programming Input "Low" Voltage		V_{MPL}	Fig. 5	–	–	1.7	V
Mode Programming Input "High" Voltage		V_{MPH}		4.0	–	–	V
$\overline{RES}$ "Low" Pulse Width		PW_{RSTL}		3.0	–	-	t_{cyc}
Mode Programming Set-up Time		t_{MPS}		2.0	–	–	t_{cyc}
Mode Programming Hold Time	$\overline{RES}$ Rise Time ≧ 1μs	t_{MPH}		0	–	–	ns
	$\overline{RES}$ Rise Time < 1μs			100	–	–	

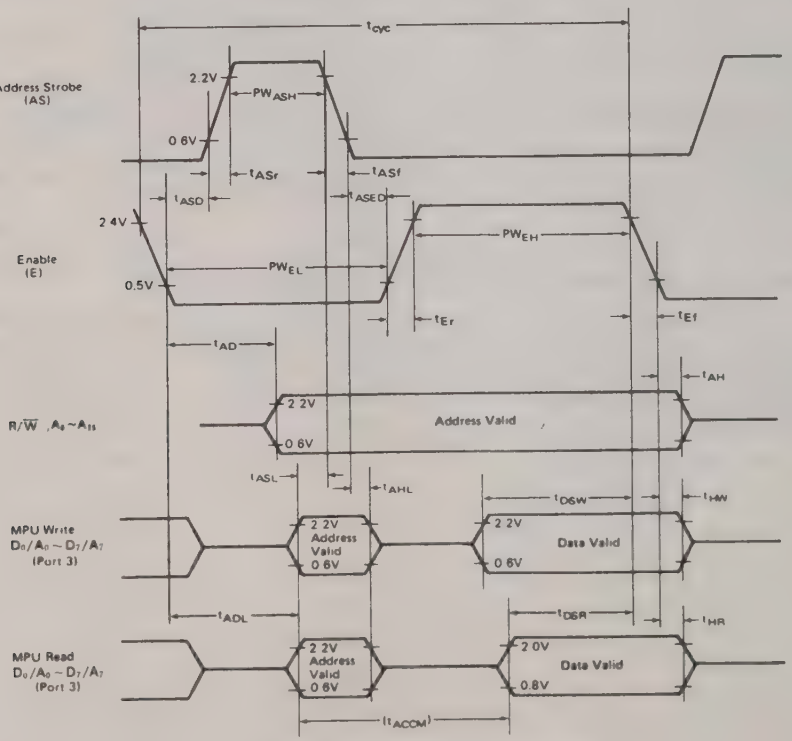


Figure 1 Expanded Multiplexed Bus Timing

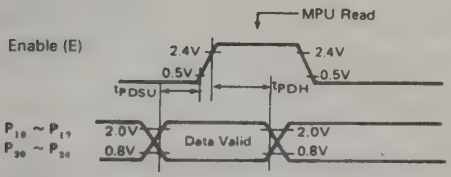


Figure 2 Data Set-up and Hold Times (MPU Read)

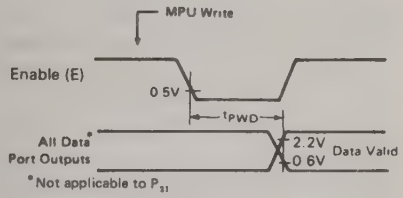


Figure 3 Port Data Delay Timing (MPU Write)

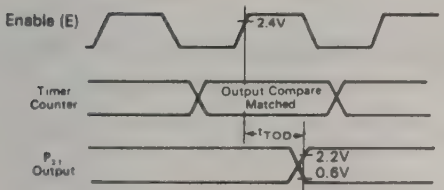


Figure 4 Timer Output Timing

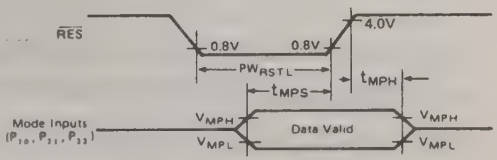


Figure 5 Mode Programming Timing

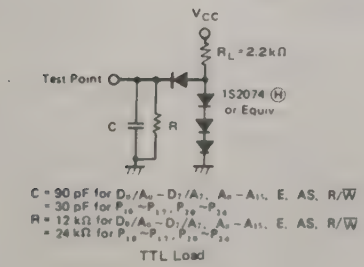

VCC
RL = 2.2kΩ
Test Point
1S2074 (H) or Equiv
C
R
C = 90 pF for D0/A8 – D7/A15, A8 – A11, E, AS, R/W
= 30 pF for P10 ~P17, P20 ~P24
R = 12 kΩ for D0/A8 ~ D7/A15, A8 – A11, E, AS, R/W
= 24 kΩ for P10 ~P17, P20 ~P24
TTL Load

Figure 6 Bus Timing Test Load

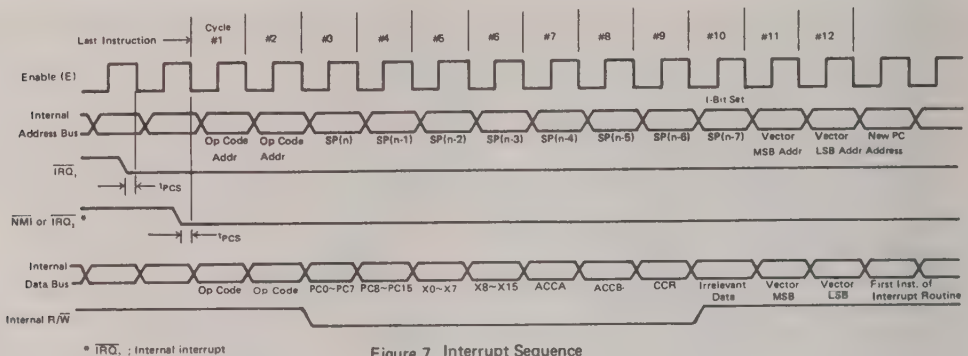


Figure 7 Interrupt Sequence

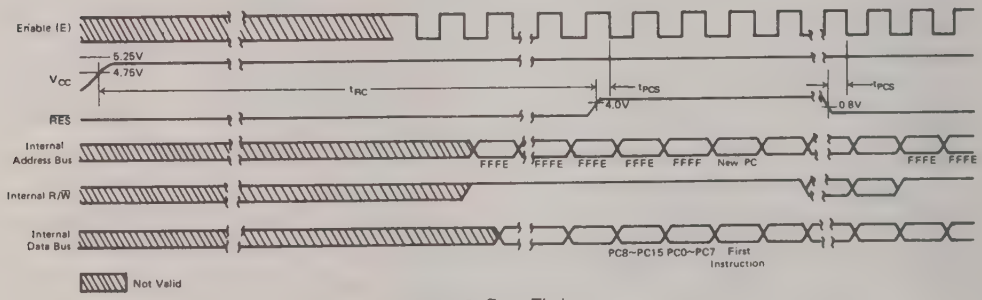


Figure 8 Reset Timing

■ SIGNAL DESCRIPTIONS

• Vcc and Vss

These two pins are used to supply power and ground to the chip. The voltage supplied will be +5 volts ±5%.

• XTAL and EXTAL

These connections are for a parallel resonant fundamental crystal, AT cut. Devide-by-4 circuitry is included with the internal clock, so a 4 MHz crystal may be used to run the system at 1 MHz. The devide-by-4 circuitry allows for use of the inexpensive 3.58 MHz Color TV crystal for non-time critical applications. Two 22pF capacitors are needed from the two crystal pins to ground to insure reliable operation. An example of the crystal interface is shown in Fig. 9. EXTAL may be driven by an external TTL compatible source with a 45% to 55% duty cycle. It will devided by 4 any frequency less than or equal to 5 MHz. XTAL must be grounded if an external clock is used.

Nominal Crystal Parameter

Item \ Crystal	4 MHz	5 MHz
C_O	7pF max.	4.7pF max.
R_S	60Ω max.	30Ω typ.

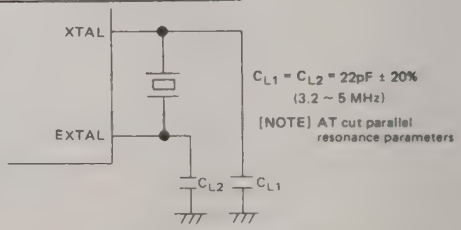


Figure 9 Crystal Interface

- **V_{CC} Standby**

This pin will supply +5 volts ±5% to the standby RAM on the chip. The first 64 bytes of RAM will be maintained in the power down mode with 8 mA current max. The circuit of figure 13 can be utilized to assure that V_{CC} Standby does not go below V_{SBB} during power down.

To retain information in the RAM during power down the following procedure is necessary:

1) Write "0" into the RAM enable bit, RAME. RAME is bit 6 of the RAM Control Register at location $0014. This disables the standby RAM, thereby protecting it at power down.
2) Keep V_{CC} Standby greater than V_{SBB}.

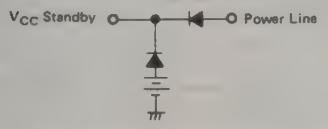


Figure 10 Battery Backup for V_{CC} Standby

- **Reset ($\overline{RES}$)**

This input is used to reset and start the MPU from a power down condition, resulting from a power failure or an initial startup of the processor. On power up, the reset must be held "Low" for at least 100 ms. When reset during operation, $\overline{RES}$ must be held "Low" at least 3 clock cycles.

When a "High" level is detected, the CPU does the following:

1) All the higher order address lines will be forced "High".
2) I/O Port 2 bits, 2, 1, and 0 are latched into programmed control bits PC2, PC1 and PC0.
3) The last two ($FFFE, $FFFF) locations in memory will be used to load the program addressed by the program counter.
4) The interrupt mask bit is set. Clear before the CPU can recognize maskable interrupts.

- **Enable (E)**

This supplies the external clock for the rest of the system when the internal oscillator is used. It is a single phase, TTL compatible clock, and will be the divide-by-4 result of the crystal oscillator frequency. It will drive one TTL load and 90 pF capacitance.

- **Non-Maskable Interrupt ($\overline{NMI}$)**

When the falling edge of the input signal is detected at this pin, the CPU begins non-maskable interrupt sequence internally. As with interrupt Request signal, the processor will complete the current instruction that is being executed before it recognizes the $\overline{NMI}$ signal. The interrupt mask bit in the Condition Code Register has no effect on $\overline{NMI}$.

In response to an $\overline{NMI}$ interrupt, the Index Register, Program Counter, Accumulators, and Condition Code Register are stored on the stack. At the end of the sequence, a 16-bit address will be loaded that points to a vectoring address located in memory locations $FFFC and $FFFD. An address loaded at these locations causes the CPU to branch to a non-maskable interrupt service routine in memory.

A 3.3 kΩ external resistor to V_{CC} should be used for wire-OR and optimum control of interrupts.

Inputs $\overline{IRQ_1}$ and $\overline{NMI}$ are hardware interrupt lines that are sampled during E and will start the interrupt routine on the $\overline{E}$ following the completion of an instruction.

- **Interrupt Request ($\overline{IRQ_1}$)**

This level sensitive input requests that an interrupt sequence be generated within the machine. The processor will complete the current instruction before it recognizes the request. At that time, if the interrupt mask bit in the Condition Code Register is not set, the machine will begin an interrupt sequence. The Index Register, Program Counter, Accumulators, and Condition Code Register are stored on the stack. Next the CPU will respond to the interrupt request by setting the interrupt mask bit "High" so that no further maskable interrupts may occur. At the end of the cycle, a 16-bit address will be loaded that points to a vectoring address which is located in memory locations $FFF8 and $FFF9. An address loaded at these locations causes the CPU to branch to an interrupt routine in memory.

The $\overline{IRQ_1}$ requires a 3.3 kΩ external resister to V_{CC} which should be used for wire-OR and optimum control of interrupts. Internal Interrupts will use an internal interrupt line ($\overline{IRQ_2}$). This interrupt will operate the same as $\overline{IRQ_1}$ except that it will use the vector address of $FFF0 through $FFF7. $\overline{IRQ_1}$ will have priority to $\overline{IRQ_2}$ if both occur at the same time. The Interrupt Mask Bit in the condition code register masks both interrupts (See Table 1).

Table 1 Interrupt Vector Location

	Vector MSB	Vector LSB	Interrupt
Highest Priority	FFFE	FFFF	$\overline{RES}$
	FFFC	FFFD	$\overline{NMI}$
	FFFA	FFFB	Software Interrupt (SWI)
	FFF8	FFF9	$\overline{IRQ_1}$
	FFF6	FFF7	ICF (Input Capture)
	FFF4	FFF5	OCF (Output Compare)
	FFF2	FFF3	TOF (Timer Overflow)
Lowest Priority	FFF0	FFF1	SCI (RDRF + ORFE + TDRE)

- **Read/Write (R/$\overline{W}$)**

This TTL compatible output signals the peripherals and memory devices whether the CPU is in a Read ("High") or a Write ("Low") state. The normal standby state of this signal is Read ("High"). This output can drive one TTL load and 90pF capacitance.

- **Address Strobe (AS)**

In the expanded multiplexed mode of operation, address strobe is output on this pin. This signal is used to latch the 8 LSB's of address which are multiplexed with data on D_0/A_0 to D_7/A_7. An 8-bit latch is utilized in conjunction with Address Strobe, as shown in figure 11. So D_0/A_0 to D_7/A_7 can become data bus during the E pulse. The timing for this signal is shown in Figure 1 of Bus Timing. This signal is also used to disable the address from the multiplexed bus allowing a deselect time, t_{ASD} before the data is enabled to the bus.

■ PORTS

There are two I/O ports on the HD6803 MPU; one 8-bit port and one 5-bit port. Each port has an associated write

only Data Direction Register which allows each I/O line to be programmed to act as an input or an output*. A "1" in the corresponding Data Direction Register bit will cause that I/O line to be an output. A "0" in the corresponding Data Direction Register bit will cause that I/O line to be an input. There are two ports: Port 1, Port 2. Their addresses and the addresses of their Data Direction registers are given in Table 2.

* The only exception is bit 1 of Port 2, which can either be data input or Timer output.

Table 2 Port and Data Direction Register Addresses

Ports	Port Address	Data Direction Register Address
I/O Port 1	$0002	$0000
I/O Port 2	$0003	$0001

- **I/O Port 1**

This is an 8-bit port whose individual bits may be defined as inputs or outputs by the corresponding bit in its data direction register. The 8 output buffers have three-state capability, allowing them to enter a high impedance state when the peripheral data lines are used as inputs. In order to be read properly, the voltage on the input lines must be greater than 2.0 V for a logic "1" and less than 0.8 V for a logic "0". As outputs, these lines are TTL compatible and may also be used as a source of up to 1 mA at 1.5 V to directly drive a Darlington base. After reset, the I/O lines are configured as inputs.

- **I/O Port 2**

This port has five lines that may be defined as inputs or outputs by its data direction register. The 5 output buffers have three-state capability, allowing them to enter a high impedance state when used as an input. In order to be read properly, the voltage on the input lines must be greater than 2.0 V for a logic "1" and less than 0.8 V for a logic "0". As outputs, this port has no internal pullup resistors but will drive TTL inputs directly. For driving CMOS inputs, external pullup resistors are required. After reset, the I/O lines are configured as inputs. Three pins on Port 2 (pin 8, 9 and 10 of the chip) are requested to set following values (Table 3) during reset. The values of above three pins during reset are latched into the three MSBs (Bit 5, 6 and 7) of Port 2 which are read only.

Port 2 can be configured as I/O and provides access to the Serial Communications Interface and the Timer. Bit 1 is the only pin restricted to data input or Timer output.

Table 3 The Values of three pins

Pin Number	Value
8	L
9	H
10	L

[NOTES] L; Logical "0"
H; Logical "1"

■ BUS

- **Data/Address Lines ($D_0/A_0 \sim D_7/A_7$)**

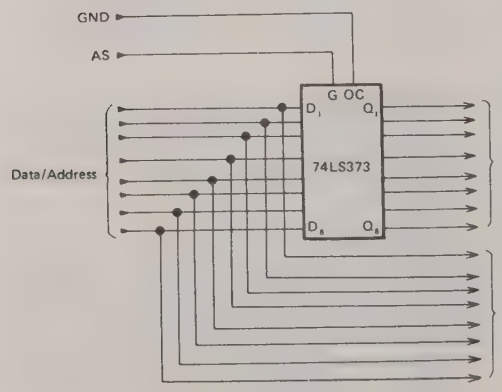
Since the data bus is multiplexed with the lower order address bus in Data/Address, latches are required to latch those address bits. The 74LS373 Transparent Octal D-type latch can be used with the HD6803 to latch the least significant address byte. Figure 11 shows how to connect the latch to the HD6803. The output control to the 74LS373 may be connected to ground.

- **Address Lines ($A_8 \sim A_{15}$)**

Each line is TTL compatible and can drive one TTL load and 90 pF. After reset, these pins become output for upper order address lines (A_8 to A_{15}).

■ INTERRUPT FLOWCHART

The Interrupt flowchart is depicted in Figure 16 and is common to every interrupt excluding reset.



Function Table

Output Control	Enable G	Enable D	Output Q
L	H	H	H
L	H	L	L
L	L	x	Q_0
H	x	x	Z

Figure 11 Latch Connection

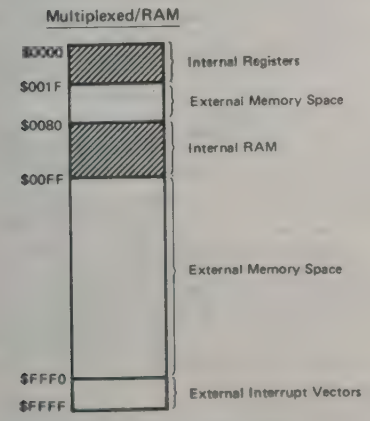
■ MEMORY MAP

The MPU can provide up to 65k byte address space. A memory map is shown in Figure 12. The first 32 locations are reserved for the MPU's internal register area, as shown in Table 4 with exceptions as indicated.

Table 4 Internal Register Area

Register	Address
Port 1 Data Direction Register**	00
Port 2 Data Direction Register**	01
Port 1 Data Register	02
Port 2 Data Register	03
Not Used	04*
Not Used	05*
Not Used	06*
Not Used	07*
Timer Control and Status Register	08
Counter (High Byte)	09
Counter (Low Byte)	0A
Output Compare Register (High Byte)	0B
Output Compare Register (Low Byte)	0C
Input Capture Register (High Byte)	0D
Input Capture Register (Low Byte)	0E
Not Used	0F*
Rate and Mode Control Register	10
Transmit/Receive Control and Status Register	11
Receive Data Register	12
Transmit Data Register	13
RAM Control Register	14
Reserved	15-1F

* External Address
** 1; Output, 0; Input



[NOTE]
Excludes the following addresses which may be used externally: $04, $05, $06, $07, and $0F.

Figure 12 HD6803 Memory Map

■ PROGRAMMABLE TIMER

The HD6803 contains an on-chip 16-bit programmable timer which may be used to measure an input waveform while independently generating an output waveform. Pulse widths for both input and output signals may vary from a few microseconds to many seconds. The timer hardware consists of

- an 8-bit control and status register,
- a 16-bit free running counter,
- a 16-bit output compare register,
- a 16-bit input capture register

A block diagram of the timer registers is shown in Figure 13.

- **Free Running Counter ($0009:$000A)**

The key element in the programmable timer is a 16-bit free running counter which is driven to increasing values by E (Enable). The counter value may be read by the CPU software at any time. The counter is cleared to zero by reset and may be considered a read-only register with one exception. Any CPU write to the counter's address ($09) will always result in preset value of $FFF8 being loaded into the counter regardless of the value involved in the write. This preset figure is intended for testing operation of the part, but may be of value in some applications.

- **Output Compare Register ($000B:$000C)**

The Output Compare Register is a 16-bit read/write register which is used to control an output waveform. The contents of this register are constantly compared with the current value of the free running counter. When a match is found, a flag is set (OCF) in the Timer Control and Status Register (TCSR) and the current value of the Output Level bit (OLVL) in the TCSR is clocked to the Output Level Register. Providing the Data Direction Register for Port 2, Bit 1 contains a "1" (Output), the output level register value will appear on the pin for Port 2 Bit 1. The values in the Output Compare Register and Output Level bit may then be changed to control the output level on the next compare value. The Output Compare Register is set to $FFFF during reset. The Compare function is inhibited for one cycle following a write to the high byte of the Output Compare Register to insure a valid 16-bit value is in the register before a compare is made.

- **Input Capture Register ($000D:$000E)**

The Input Capture Register is a 16-bit read-only register used to store the current value of the free running counter when the proper transition of an external input signal occurs. The input transition change required to trigger the counter transfer is controlled by the input Edge bit (IEDG) in the TCSR. The Data Direction Register bit for Port 2 Bit 0, should* be clear (zero) in order to gate in the external input signal to the edge detect unit in the timer.

The input pulse width must be at least two E-cycles to ensure an input capture under all conditions.

* With Port 2 Bit 0 configured as an output and set to "1", the external input will still be seen by the edge detect unit.

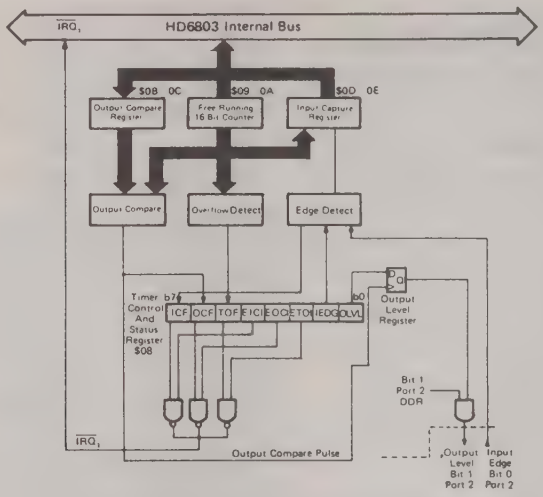


Figure 13 Block Diagram of Programmable Timer

Timer Control and Status Register

7	6	5	4	3	2	1	0	
ICF	OCF	TOF	EICI	EOCI	ETOI	IEDG	OLVL	$0008

- **Timer Control and Status Register (TCSR) ($0008)**

The Timer Control and Status Register consists of an 8-bit register of which all 8 bits are readable but only the low order 5 bits may be written. The upper three bits contain read-only timer status information and indicate the followings:

- a proper transition has taken place on the input pin with a subsequent transfer of the current counter value to the input capture register.
- a match has been found between the value in the free running counter and the output compare register, and when $0000 is in the free running counter.

Each of the flags may be enabled onto the HD6803 internal bus ($\overline{IRQ_2}$) with an individual Enable bit in the TCSR. If the I-bit in the HD6803 Condition Code register has been cleared, a prior vectored interrupt will occur corresponding to the flag bit(s) set. A description for each bit follows:

Bit 0 **OLVL** Output Level – This value is clocked to the output level register on a successful output compare. If the DDR for Port 2 bit 1 is set, the value will appear on the output pin.

Bit 1 **IEDG** Input Edge – This bit controls which transition of an input will trigger a transfer of the counter to the input capture register. The DDR for Port 2 Bit 0 must be clear for this function to operate. IEDG = 0 Transfer takes place on a negative edge ("High"-to-"Low" transition).
IEDG = 1 Transfer takes place on a positive edge ("Low"-to-"High" transition).

Bit 2 **ETOI** Enable Timer Overflow Interrupt – When set, this bit enables $\overline{IRQ_2}$ to occur on the internal bus for a TOF interrupt; when clear the interrupt is inhibited.

Bit 3 **EOCI** Enable Output Compare Interrupt – When set, this bit enables $\overline{IRQ_2}$ to appear on the internal bus for an output compare interrupt; when clear the interrupt is inhibited.

Bit 4 **EICI** Enable input Capture Interrupt – When set, this bit enables $\overline{IRQ_2}$ to occur on the internal bus for an input capture interrupt; when clear the interrupt is inhibited.

Bit 5 **TOF** Timer Overflow Flag – This read-only bit is set when the counter contains $FFFF. It is cleared by a read of the TCSR (with TOF set) followed by an CPU read of the Counter ($09).

Bit 6 **OCF** Output Compare Flag – This read-only bit is set when a match is found between the output compare register and the free running counter. It is cleared by a read of the TCSR (with OCF set) followed by an CPU write to the output compare register ($0B or $0C).

Bit 7 **ICF** Input Capture Flag – This read-only status bit is set by a proper transition on the input; it is cleared by a read of the TCSR (with ICF set) followed by an CPU read of the Input Capture Register ($0D).

■ SERIAL COMMUNICATIONS INTERFACE

The HD6803 contains a full-duplex asynchronous serial communications interface (SCI) on chip. The controller comprises a transmitter and a receiver which operate independently or each other but in the same data format and at the same data rate. Both transmitter and receiver communicate with the CPU via the data bus and with the outside world via pins 2, 3, and 4 of Port 2. The hardware, software, and registers are explained in the following paragraphs.

• Wake-Up Feature

In a typical multi-processor application, the software protocol will usually contain a destination address in the initial byte(s) of the message. In order to permit non-selected MPU's to ignore the remainder of the message, a wake-up feature is included whereby all further interrupt processing may be optionally inhibited until the beginning of the next message. When the next message appears, the hardware re-enables (or "wakes-up") for the next message. The "wake-up" is automatically triggered by a string of ten consecutive 1's which indicates an idle transmit line. The software protocol must provide for the short idle period between any two consecutive messages.

• Programmable Options

The following features of the HD6803 serial I/O section are programmable:

- format – standard mark/space (NRZ)
- Clock – external or internal
- baud rate – one of 4 per given CPU ϕ_2 clock frequency or external clock ×8 input
- wake-up feature – enabled or disabled
- Interrupt requests – enabled or masked individually for transmitter and receiver data registers
- clock output – internal clock enabled or disabled to Port 2 (Bit 2)
- Port 2 (bits 3 and 4) – dedicated or not dedicated to serial I/O individually for transmitter and receiver.

• Serial Communications Hardware

The serial communications hardware is controlled by 4 registers as shown in Figure 14. The registers include:

- an 8-bit control and status register
- a 4-bit rate and mode control register (write only)
- an 8-bit read only receive data register and
- an 8-bit write only transmit data register.

In addition to the four registers, the serial I/O section utilizes bit 3 (serial input) and bit 4 (serial output) of Port 2. Bit 2 of Port 2 is utilized if the internal-clock-out or external-clock-in options are selected.

Transmit/Receive Control and Status (TRCS) Register

The TRCS register consists of an 8-bit register of which all 8 bits may be read while only bits 0~4 may be written. The register is initialized to $20 by reset. The bits in the TRCS register are defined as follows:

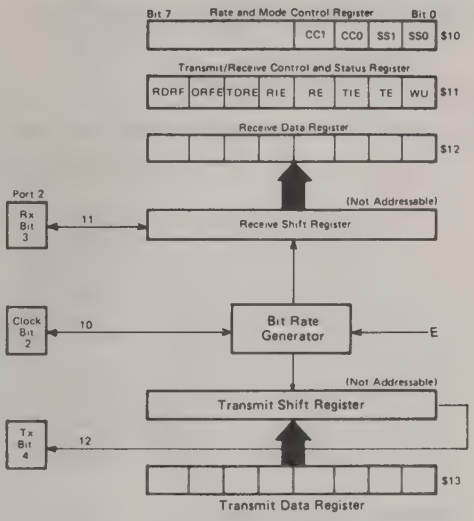


Figure 14 Serial I/O Registers

Bit 0 **WU** "Wake-up" on Next Message – set by HD6803 software and cleared by hardware on receipt of ten consecutive 1's or reset of RE flag. It should be noted that RE flag should be set in advance of CPU set of WU flag.

Bit 1 **TE** Transmit Enable – set by HD6803 to produce preamble of nine consecutive 1's and to enable gating of transmitter output to Port 2, bit 4 regardless of the DDR value corresponding to this bit; when clear, serial I/O has no effect on Port 2 bit 4.

TE set should be after at least one bit time of data transmit rate from the set-up of transmit data rate and mode.

Bit 2 **TIE** Transmit Interrupt Enable – when set, will permit an $\overline{IRQ_2}$ interrupt to occur when bit 5 (TDRE) is set; when clear, the TDRE value is masked from the bus.

Bit 3 **RE** Receiver Enable – when set, gates Port 2 bit 3 to input of receiver regardless of DDR value for this bit; when clear, serial I/O has no effect on Port 2 bit 3.

Bit 4 **RIE** Receiver Interrupt Enable – when set, will permit an $\overline{IRQ_2}$ interrupt to occur when bit 7 (RDRF) or bit 6 (ORFE) is set; when clear, the interrupt is masked.

Transmit/Receive Control and Status Register

7	6	5	4	3	2	1	0	
RDRF	ORFE	TDRE	RIE	RE	TIE	TE	WU	ADDR $0011

Bit 5 **TDRE** Transmit Data Register Empty – set by hardware when a transfer is made from the transmit data register to the output shift register. The TDRE bit is cleared by reading the status register, then writing a new byte into the transmit data register, TDRE is initialized to 1 by reset.

Bit 6 **ORFE** Over-Run-Framing Error – set by hardware when an overrun or framing error occurs (receive only).

Rate and Mode Control Register

7	6	5	4	3	2	1	0
X	X	X	X	CC1	CC0	SS1	SS0

ADDR : $0010

An overrun is defined as a new byte received with last byte still in Data Register/Buffer. A framing error has occured when the byte boundaries in bit stream are not synchronized to bit counter. If WU-flag is set, the ORFE bit will not be set. The ORFE bit is cleard by reading the status register, then reading the Receive Data Register, or by reset.

Bit 7 **RDRF** Receiver Data Register Full-set by hardware when a transfer from the input shift register to the receiver data register is made. If WU-flag is set, the RDRF bit will not be set. The RDRF bit is cleared by reading the status register, then reading the Receive Data Register, or by reset.

Rate and Mode Control Register (RMCR)

The Rate and Mode Control register controls the following serial I/O variables:

- Baud rate
- format
- clocking source,
- Port 2 bit 2 configuration

The register consists of 4 bits all of which are write-only and cleared by reset. The 4 bits in the register may be considered as a pair of 2-bit fields. The two low order bits control the bit rate for internal clocking and the remaining two bits control the format and clock select logic. The register definition is as follows:

Bit 0 **SS0**, Bit 1 **SS1** } Speed Select – These bits select the Baud rate for the internal clock. The four rates which may be selected are a function of the CPU ϕ_2 clock frequency. Table 5 lists the available Baud rates.

Bit 2 **CC0**, Bit 3 **CC1** } Clock Control and Format Select – this 2-bit field controls the format and clock select logic. Table 6 defines the bit field.

Table 5 SCI Bit Times and Rates

	XTAL	2.4576 MHz	4.0 MHz	4.9152 MHz*
SS1 : SS0	E	614.4 kHz	1.0 MHz	1.2288 MHz
0 0	E ÷ 16	26 μs/38,400 Baud	16 μs/62,500 Baud	13.0 μs/76,800 Baud
0 1	E ÷ 128	208 μs/4,800 Baud	128 μs/7812.5 Baud	104.2 μs/9,600 Baud
1 0	E ÷ 1024	1.67 ms/600 Baud	1.024 ms/976.6 Baud	833.3 μs/1,200 Baud
1 1	E ÷ 4096	6.67 ms/150 Baud	4.096 ms/244.1 Baud	3.33 ms/300 Baud

* HD6803-1 Only

Table 6 SCI Format and Clock Source Control

CC1, CC0	Format	Clock Source	Port 2 Bit 2	Port 2 Bit 3	Port 2 Bit 4
0 0	–	–	–	–	–
0 1	NRZ	Internal	Not Used	••	••
1 0	NRZ	Internal	Output*	••	••
1 1	NRZ	External	Input	••	••

* Clock output is available regardless of values for bits RE and TE.
•• Bit 3 is used for serial input if RE = "1" in TRCS; bit 4 is used for serial output if TE = "1" in TRCS.

Internally Generated Clock

If the user wishes for the serial I/O to furnish a clock, the following requirements are applicable:

- the values of RE and TE are immaterial.
- CC1, CC0 must be set to 10
- the maximum clock rate will be E ÷ 16.
- the clock will be at 1x the bit rate and will have a rising edge at mid-bit.

Externally Generated Clock

If the user wishes to provide an external clock for the serial I/O, the following requirements are applicable:

- the CC1, CC0, field in the Rate and Mode Control Register must be set to 11,
- the external clock must be set to 8 times (×8) the desired baud rate and
- the maximum external clock frequency is 1.0 MHz.

• Serial Operations

The serial I/O hardware should be initialized by the HD6803 software prior to operation. This sequence will normally consist of;

- writing the desired operation control bits to the Rate and Mode Control Register and
- writing the desired operational control bits in the Transmit/Receive Control and Status Register.

The Transmitter Enable (TE) and Receiver Enable (RE) bits may be left set for dedicated operations.

Transmit Operations

The transmit operation is enabled by the TE bit in the Transmit/Receive Control and Status Register. This bit when set, gates the output of the serial transmit shift register to Port 2 Bit 4 and takes unconditional control over the Data Direction Register value for Port 2, Bit 4.

Following a $\overline{RES}$ the user should configure both the Rate and Mode Control Register and the Transmit/Receive Control and Status Register for desired operation. Setting the TE bit during this procedure initiates the serial output by first transmitting a nine-bit preamble of 1's. Following the preamble, internal synchronization is established and the transmitter section is ready for operation.

At this point one of two situation exist:

1) if the Transmit Data Register is empty (TDRE = 1), a continuous string of ones will be sent indicating an idle line, or,
2) if data has been loaded into the Transmit Data Register (TDRE = 0), the word is transferred to the output shift register and transmission of the data word will begin.

During the data transmit, the 0 start bit is first transmitted. Then the 8 data bits (beginning with bit 0) followed by the stop bit, are transmitted. When the Transmitter Data Register has been emptied, the hardware sets the TDRE flag bit.

If the HD6803 fails to respond to the flag within the proper time, (TDRE is still set when the next normal transfer from the parallel data register to the serial output register should occur) then a 1 will be sent (instead of a 0) at "Start" bit time, followed by more 1's until more data is supplied to the data register. No 0's will be sent while TDRE remains a 1.

Receive Operation

The receive operation is enabled by the RE bit which gates in the serial input through Port 2, Bit 3. The receiver section operation is conditioned by the contents of the Transmit/Receive Control and Status Register and the Rate and Mode Control Register.

The receiver bit interval is divided into 8 sub-intervals for internal synchronization. In the NRZ Mode, the received bit stream is synchronized by the first 0 (space) encountered.

The approximate center of each bit time is strobed during the next 10 bits. If the tenth bit is not a 1 (stop bit) a framing error is assumed, and bit ORFE is set. If the tenth bit as a 1, the data is transferred to the Receive Data Register, and interrupt flag RDRF is set. If RDRF is still set at the next tenth bit time, ORFE will be set, indicating an overrun has occurred. When the HD6803 responds to either flag (RDRF or ORFE) by reading the status register followed by reading the Data Register, RDRF (or ORFE) will be cleared.

■ RAM CONTROL REGISTER

This register, which is addressed at $0014, gives status information about the standby RAM. A 0 in the RAM enable bit (RAME) will disable the standby RAM, thereby protecting it at power down if V_{CC} Standby is held greater than V_{SBB} volts, as explained previously in the signal description for V_{CC} Standby.

RAM Control Register

$0014	STBY PWR	RAME	X	X	X	X	X	X

Bit 0 Not used.
Bit 1 Not used.
Bit 2 Not used.
Bit 3 Not used.
Bit 4 Not used.
Bit 5 Not used.
Bit 6 **RAME** The RAM Enable control bit allows the user the ability to disable the standby RAM. This bit is set to a logic "1" by $\overline{RES}$ which enables the standby RAM and can be written to one or zero under program control. When the RAM is disabled, data is read from external memory.
Bit 7 **STBY PWR** The Standby Power bit is cleared when the standby voltage is removed. This bit is a read/write status flag that the user can read which indicates that the standby RAM voltage has been applied, and the data in the standby RAM is valid.

■ GENERAL DESCRIPTION OF INSTRUCTION SET

The HD6803 is upward object code compatible with the HD6800 as it implements the full HMCS6800 instruction set. The execution times of key instructions have been reduced to increase throughout. In addition, new instructions have been added; these include 16-bit operations and a hardware multiply.

Included in the instruction set section are the following:

- CPU Programming Model—Figure 15.
- Addressing modes
- Accumulator and memory instructions – Table 7
- New instructions
- Index register and stack manipulations instructions – Table 8
- Jump and branch instructions – Table 9
- Condition code register manipulation instructions - Table 10
- Instructions Execution times in machine cycles - Table 11
- Summary of cycle by cycle operation – Table 12
- Summary of undefined instructions – Table 13

• CPU Programming Model

The programming model for the HD6803 is shown in Figure 15. The double (D) accumulator is physically the same as the Accumulator A concatenated with the Accumulator B so that any operation using accumulator D will destroy information in A and B.

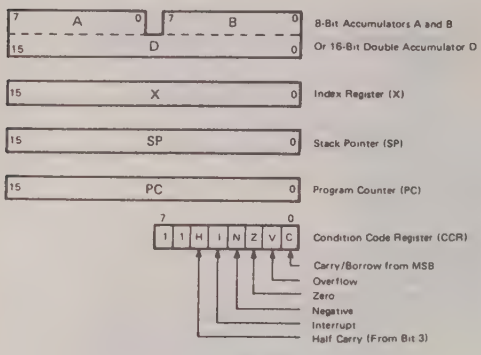


Figure 15 CPU Programming Model

- **CPU Addressing Modes**

The HD6803 8-bit micro processing unit has seven address modes that can be used by a programmer, with the addressing mode a function of both the type of instruction and the coding within the instruction. A summary of the addressing modes for a particular instruction can be found in Table 11 along with the associated instruction execution time that is given in machine cycles. With a clock frequency of 4 MHz, these times would be microseconds.

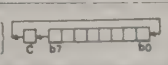
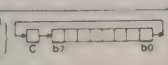
Accumulator (ACCX) Addressing

In accumulator only addressing, either accumulator A or accumulator B is specified. These are one-byte instructions.

Immediate Addressing

In immediate addressing, the operand is contained in the second byte of the instruction except LDS and LDX which have the operand in the second and third bytes of the instruction. The CPU addresses this location when it fetches the immediate instruction for execution. These are two or three-byte instructions.

Table 7 Accumulator & Memory Instructions

Operations	Mnemonic	Addressing Modes															Boolean/ Arithmetic Operation	Condition Code Register					
		IMMED.			DIRECT			INDEX			EXTEND			IMPLIED				5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#		H	I	N	Z	V	C
Add	ADDA	8B	2	2	9B	3	2	AB	4	2	BB	4	3				A + M → A	↕	•	↕	↕	↕	↕
	ADDB	CB	2	2	DB	3	2	EB	4	2	FB	4	3				B + M → B	↕	•	↕	↕	↕	↕
Add Double	ADDD	C3	4	3	D3	5	2	E3	6	2	F3	6	3				A B + M M + 1 → A B	•	•	↕	↕	↕	↕
Add Accumulators	ABA													1B	2	1	A + B → A	↕	•	↕	↕	↕	↕
Add With Carry	ADCA	89	2	2	99	3	2	A9	4	2	B9	4	3				A + M + C → A	↕	•	↕	↕	↕	↕
	ADCB	C9	2	2	D9	3	2	E9	4	2	F9	4	3				B + M + C → B	↕	•	↕	↕	↕	↕
AND	ANDA	84	2	2	94	3	2	A4	4	2	B4	4	3				A·M → A	•	•	↕	↕	R	•
	ANDB	C4	2	2	D4	3	2	E4	4	2	F4	4	3				B·M → B	•	•	↕	↕	R	•
Bit Test	BIT A	85	2	2	95	3	2	A5	4	2	B5	4	3				A·M	•	•	↕	↕	R	•
	BIT B	C5	2	2	D5	3	2	E5	4	2	F5	4	3				B·M	•	•	↕	↕	R	•
Clear	CLR							6F	6	2	7F	6	3				00 → M	•	•	R	S	R	R
	CLRA													4F	2	1	00 → A	•	•	R	S	R	R
	CLRB													5F	2	1	00 → B	•	•	R	S	R	R
Compare	CMPA	81	2	2	91	3	2	A1	4	2	B1	4	3				A - M	•	•	↕	↕	↕	↕
	CMPB	C1	2	2	D1	3	2	E1	4	2	F1	4	3				B - M	•	•	↕	↕	↕	↕
Compare Accumulators	CBA													11	2	1	A - B	•	•	↕	↕	↕	↕
Complement, 1's	COM							63	6	2	73	6	3				$\overline{M}$ → M	•	•	↕	↕	R	S
	COMA													43	2	1	$\overline{A}$ → A	•	•	↕	↕	R	S
	COMB													53	2	1	$\overline{B}$ → B	•	•	↕	↕	R	S
Complement, 2's (Negate)	NEG							60	6	2	70	6	3				00 - M → M	•	•	↕	↕	①	②
	NEGA													40	2	1	00 - A → A	•	•	↕	↕	①	②
	NEGB													50	2	1	00 - B → B	•	•	↕	↕	①	②
Decimal Adjust, A	DAA													19	2	1	Converts binary add of BCD characters into BCD format	•	•	↕	↕	↕	③
Decrement	DEC							6A	6	2	7A	6	3				M - 1 → M	•	•	↕	↕	④	•
	DECA													4A	2	1	A - 1 → A	•	•	↕	↕	④	•
	DECB													5A	2	1	B - 1 → B	•	•	↕	↕	④	•
Exclusive OR	EORA	88	2	2	98	3	2	A8	4	2	B8	4	3				A ⊕ M → A	•	•	↕	↕	R	•
	EORB	C8	2	2	D8	3	2	E8	4	2	F8	4	3				B ⊕ M → B	•	•	↕	↕	R	•
Increment	INC							6C	6	2	7C	6	3				M + 1 → M	•	•	↕	↕	⑤	•
	INCA													4C	2	1	A + 1 → A	•	•	↕	↕	⑤	•
	INCB													5C	2	1	B + 1 → B	•	•	↕	↕	⑤	•
Load Accumulator	LDAA	86	2	2	96	3	2	A6	4	2	B6	4	3				M → A	•	•	↕	↕	R	•
	LDAB	C6	2	2	D6	3	2	E6	4	2	F6	4	3				M → B	•	•	↕	↕	R	•
Load Double Accumulator	LDD	CC	3	3	DC	4	2	EC	5	2	FC	5	3				M + 1 → B, M → A	•	•	↕	↕	R	•
Multiply Unsigned	MUL													3D	10	1	A × B → A B	•	•	•	•	•	⑬
OR, Inclusive	ORAA	8A	2	2	9A	3	2	AA	4	2	BA	4	3				A + M → A	•	•	↕	↕	R	•
	ORAB	CA	2	2	DA	3	2	EA	4	2	FA	4	3				B + M → B	•	•	↕	↕	R	•
Push Data	PSHA													36	3	1	A → Msp, SP - 1 → SP	•	•	•	•	•	•
	PSHB													37	3	1	B → Msp, SP - 1 → SP	•	•	•	•	•	•
Pull Data	PULA													32	4	1	SP + 1 → SP, Msp → A	•	•	•	•	•	•
	PULB													33	4	1	SP + 1 → SP, Msp → B	•	•	•	•	•	•
Rotate Left	ROL							69	6	2	79	6	3					•	•	↕	↕	⑥	↕
	ROLA													49	2	1		•	•	↕	↕	⑥	↕
	ROLB													59	2	1		•	•	↕	↕	⑥	↕
Rotate Right	ROR							66	6	2	76	6	3					•	•	↕	↕	⑥	↕
	RORA													46	2	1		•	•	↕	↕	⑥	↕
	RORB													56	2	1		•	•	↕	↕	⑥	↕

The Condition Code Register notes are listed after Table 10.

(Continued)

Table 7 Accumulator & Memory Instructions (Continued)

Operations	Mnemonic	Addressing Modes															Boolean/ Arithmetic Operation	Condition Code Register					
		IMMED.			DIRECT			INDEX			EXTEND			IMPLIED				5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#		H	I	N	Z	V	C
Shift Left Arithmetic	ASL							68	6	2	78	6	3				M ← 0	•	•	↕	↕	⑥	↕
	ASLA													48	2	1	A ← 0	•	•	↕	↕	⑥	↕
	ASLB													58	2	1	B ← 0	•	•	↕	↕	⑥	↕
Double Shift Left, Arithmetic	ASLD													05	3	1	ACC A/ ACC B ← 0	•	•	↕	↕	⑥	↕
Shift Right Arithmetic	ASR							67	6	2	77	6	3				M →	•	•	↕	↕	⑥	↕
	ASRA													47	2	1	A →	•	•	↕	↕	⑥	↕
	ASRB													57	2	1	B →	•	•	↕	↕	⑥	↕
Shift Right Logical	LSR							64	6	2	74	6	3				M → 0	•	•	R	↕	⑥	↕
	LSRA													44	2	1	A → 0	•	•	R	↕	⑥	↕
	LSRB													54	2	1	B → 0	•	•	R	↕	⑥	↕
Double Shift Right Logical	LSRD													04	3	1	0 → ACC A/ ACC B → C	•	•	R	↕	⑥	↕
Store Accumulator	STAA				97	3	2	A7	4	2	B7	4	3				A → M	•	•	↕	↕	R	•
	STAB				D7	3	2	E7	4	2	F7	4	3				B → M	•	•	↕	↕	R	•
Store Double Accumulator	STD				DD	4	2	ED	5	2	FD	5	3				A → M B → M + 1	•	•	↕	↕	R	•
Subtract	SUBA	80	2	2	90	3	2	A0	4	2	B0	4	3				A − M → A	•	•	↕	↕	↕	↕
	SUBB	C0	2	2	D0	3	2	E0	4	2	F0	4	3				B − M → B	•	•	↕	↕	↕	↕
Double Subtract	SUBD	83	4	3	93	5	2	A3	6	2	B3	6	3				A . B − M M + 1 → A B	•	•	↕	↕	↕	↕
Subtract Accumulators	SBA													10	2	1	A − B → A	•	•	↕	↕	↕	↕
Subtract With Carry	SBCA	82	2	2	92	3	2	A2	4	2	B2	4	3				A − M − C → A	•	•	↕	↕	↕	↕
	SBCB	C2	2	2	D2	3	2	E2	4	2	F2	4	3				B − M − C → B	•	•	↕	↕	↕	↕
Transfer Accumulators	TAB													16	2	1	A → B	•	•	↕	↕	R	•
	TBA													17	2	1	B → A	•	•	↕	↕	R	•
Test Zero or Minus	TST							6D	6	2	7D	6	3				M − 00	•	•	↕	↕	R	R
	TSTA													4D	2	1	A − 00	•	•	↕	↕	R	R
	TSTB													5D	2	1	B − 00	•	•	↕	↕	R	R

The Condition Code Register notes are listed after Table 10.

Direct Addressing

In direct addressing, the address of the operand is contained in the second byte of the instruction. Direct addressing allows the user to directly address the lowest 256 bytes in the machine i.e., locations zero through 255. Enhanced execution times are achieved by storing data in these locations. In most configurations, it should be a random access memory. These are two-byte instructions.

Extended Addressing

In extended addressing, the address contained in the second byte of the instruction is used as the higher 8-bits of the address of the operand. The third byte of the instruction is used as the lower 8-bits of the address for the operand. This is an absolute address in memory. These are three-byte instructions.

Indexed Addressing

In indexed addressing, the address contained in the second byte of the instruction is added to the index register's lowest 8-bits in the CPU. The carry is then added to the higher order 8-bits of the index register. This result is then used to address memory. The modified address is held in a temporary address register so there is no change to the index register. These are two-byte instructions.

Implied Addressing

In the implied addressing mode the instruction gives the address (i.e., stack pointer, index register, etc.). These are one-byte instructions.

Relative Addressing

In relative addressing, the address contained in the second byte of the instruction is added to the program counter's lowest 8-bits plus two. The carry or borrow is then added to the high 8-bits. This allows the user to address data within a range of -126 to +129 bytes of the present instruction. These are two-byte instructions.

- **New Instructions**

In addition to the existing 6800 Instruction Set, the following new instructions are incorporated in the HD6803 Microcomputer.

ABX Adds the 8-bit unsigned accumulator B to the 16-bit X-Register taking into account the possible carry out of the low order byte of the X-Register.

ADDD Adds the double precision ACCD* to the double precision value M:M+1 and places the results in ACCD.

ASLD Shifts all bits of ACCD one place to the left. Bit 0 is loaded with zero. The C bit is loaded from the most significant bit of ACCD.

LDD Loads the contents of double precision memory location into the double accumulator A:B. The condition codes are set according to the data.

LSRD Shifts all bits of ACCD one place to the right. Bit 15 is loaded with zero. The C bit is loaded from the least significant bit to ACCD.

MUL Multiplies the 8 bits in accumulator A with the 8 bits in accumulator B to obtain a 16-bit unsigned number in A:B, ACCA contains MSB of result.

PSHX The contents of the index register is pushed onto the stack at the address contained in the stack pointer. The stack pointer is decremented by 2.

PULX The index register is pulled from the stack beginning at the current address contained in the stack pointer +1. The stack pointer is incremented by 2 in total.

STD Stores the contents of double accumulator A:B in memory. The contents of ACCD remain unchanged.

SUBD Subtracts the contents of M:M + 1 from the contents of double accumulator AB and places the result in ACCD.

BRN Never branches. If effect, this instruction can be considered a two byte NOP (No operation) requiring three cycles for execution.

CPX Internal processing modified to permit its use with any conditional branch instruction.

*ACCD is the 16 bit register (A:B) formed by concatenating the A and B accumulators. The A-accumulator is the most significant byte.

Table 8 Index Register and Stack Manipulation Instructions

Pointer Operations	Mnemonic	Addressing Modes																Boolean/ Arithmetic Operation	Condition Code Register					
		IMMED.			DIRECT			INDEX			EXTND			IMPLIED				5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#		H	I	N	Z	V	C
Compare Index Reg	CPX	8C	4	3	9C	5	2	AC	6	2	BC	6	3					X − M : M + 1	•	•	↕	↕	↕	↕
Decrement Index Reg	DEX													09	3	1		X − 1 → X	•	•	•	↕	•	•
Decrement Stack Pntr	DES													34	3	1		SP − 1 → SP	•	•	•	•	•	•
Increment Index Reg	INX													08	3	1		X + 1 → X	•	•	•	↕	•	•
Increment Stack Pntr	INS													31	3	1		SP + 1 → SP	•	•	•	•	•	•
Load Index Reg	LDX	CE	3	3	DE	4	2	EE	5	2	FE	5	3					M → X_H, (M + 1) → X_L	•	•	⑦	↕	R	•
Load Stack Pntr	LDS	8E	3	3	9E	4	2	AE	5	2	BE	5	3					M → SP_H, (M + 1) → SP_L	•	•	⑦	↕	R	•
Store Index Reg	STX				DF	4	2	EF	5	2	FF	5	3					X_H → M, X_L → (M + 1)	•	•	⑦	↕	R	•
Store Stack Pntr	STS				9F	4	2	AF	5	2	BF	5	3					SP_H → M, SP_L → (M + 1)	•	•	⑦	↕	R	•
Index Reg → Stack Pntr	TXS													35	3	1		X − 1 → SP	•	•	•	•	•	•
Stack Pntr → Index Reg	TSX													30	3	1		SP + 1 → X	•	•	•	•	•	•
Add	ABX													3A	3	1		B + X → X	•	•	•	•	•	•
Push Data	PSHX													3C	4	1		X_L → M_{SP}, SP − 1 → SP X_H → M_{SP}, SP − 1 → SP	•	•	•	•	•	•
Pull Data	PULX													38	5	1		SP + 1 → SP, M_{SP} → X_H SP + 1 → SP, M_{SP} → X_L	•	•	•	•	•	•

The Condition Code Register notes are listed after Table 10.

Table 9 Jump and Branch Instructions

Operations	Mnemonic	Addressing Modes														Branch Test	Condition Code Register					
		RELATIVE			DIRECT			INDEX			EXTND			IMPLIED				5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#		H	I	N	Z	V	C
Branch Always	BRA	20	3	2													None	•	•	•	•	•	•
Branch Never	BRN	21	3	2													None	•	•	•	•	•	•
Branch If Carry Clear	BCC	24	3	2													C = 0	•	•	•	•	•	•
Branch If Carry Set	BCS	25	3	2													C = 1	•	•	•	•	•	•
Branch If = Zero	BEQ	27	3	2													Z = 1	•	•	•	•	•	•
Branch If ≥ Zero	BGE	2C	3	2													N ⊕ V = 0	•	•	•	•	•	•
Branch If > Zero	BGT	2E	3	2													Z + (N ⊕ V) = 0	•	•	•	•	•	•
Branch If Higher	BHI	22	3	2													C + Z = 0	•	•	•	•	•	•
Branch If ≤ Zero	BLE	2F	3	2													Z + (N ⊕ V) = 1	•	•	•	•	•	•
Branch If Lower Or Same	BLS	23	3	2													C + Z = 1	•	•	•	•	•	•
Branch If < Zero	BLT	2D	3	2													N ⊕ V = 1	•	•	•	•	•	•
Branch If Minus	BMI	2B	3	2													N = 1	•	•	•	•	•	•
Branch If Not Equal Zero	BNE	26	3	2													Z = 0	•	•	•	•	•	•
Branch If Overflow Clear	BVC	28	3	2													V = 0	•	•	•	•	•	•
Branch If Overflow Set	BVS	29	3	2													V = 1	•	•	•	•	•	•
Branch If Plus	BPL	2A	3	2													N = 0	•	•	•	•	•	•
Branch To Subroutine	BSR	8D	6	2														•	•	•	•	•	•
Jump	JMP							6E	3	2	7E	3	3					•	•	•	•	•	•
Jump To Subroutine	JSR				9D	5	2	AD	6	2	BD	6	3					•	•	•	•	•	•
No Operation	NOP													01	2	1	Advances Prog. Cntr. Only	•	•	•	•	•	•
Return From Interrupt	RTI													3B	10	1		——			⑧	——	
Return From Subroutine	RTS													39	5	1		•	•	•	•	•	•
Software Interrupt	SWI													3F	12	1		•	S	•	•	•	•
Wait for Interrupt	WAI													3E	9	1		•	⑨	•	•	•	•

Table10 Condition Code Register Manipulation Instructions

Operations	Mnemonic	Addressing Modes			Boolean Operation	Condition Code Register					
		IMPLIED				5	4	3	2	1	0
		OP	~	#		H	I	N	Z	V	C
Clear Carry	CLC	0C	2	1	0 → C	•	•	•	•	•	R
Clear Interrupt Mask	CLI	0E	2	1	0 → I	•	R	•	•	•	•
Clear Overflow	CLV	0A	2	1	0 → V	•	•	•	•	R	•
Set Carry	SEC	0D	2	1	1 → C	•	•	•	•	•	S
Set Interrupt Mask	SEI	0F	2	1	1 → I	•	S	•	•	•	•
Set Overflow	SEV	0B	2	1	1 → V	•	•	•	•	S	•
Accumulator A → CCR	TAP	06	2	1	A → CCR	——	——	——	⑩	——	——
CCR → Accumulator A	TPA	07	2	1	CCR → A	•	•	•	•	•	•

Condition Code Register Notes: (Bit set it test is true and cleared otherwise)

① (Bit V) Test: Result = 10000000?
② (Bit C) Test: Result = 00000000?
③ (Bit C) Test: Decimal value of most significant BCD Character greater than nine? (Not cleared if previously set)
④ (Bit V) Test: Operand = 10000000 prior to execution?
⑤ (Bit V) Test: Operand = 01111111 prior to execution?
⑥ (Bit V) Test: Set equal to result of N ⊕ C after shift has occurred.
⑦ (Bit N) Test: Result less than zero? (Bit 15 = 1)
⑧ (All) Load Condition Code Register from Stack. (See Special Operations)
⑨ (Bit I) Set when interrupt occurs. If previously set, a Non-Maskable Interrupt is required to exit the wait state.
⑩ (All) Set according to the contents of Accumulator A.
⑪ (Bit C) Set equal to result of Bit 7 (ACCB)

Table 11 Instruction Execution Times in Machine Cycle

	ACCX	Immediate	Direct	Extended	Indexed	Implied	Relative		ACCX	Immediate	Direct	Extended	Indexed	Implied	Relative
ABA	•	•	•	•	•	2	•	INX	•	•	•	•	•	3	•
ABX	•	•	•	•	•	3	•	JMP	•	•	•	3	3	•	•
ADC	•	2	3	4	4	•	•	JSR	•	•	5	6	6	•	•
ADD	•	2	3	4	4	•	•	LDA	•	2	3	4	4	•	•
ADDD	•	4	5	6	6	•	•	LDD	•	3	4	5	5	•	•
AND	•	2	3	4	4	•	•	LDS	•	3	4	5	5	•	•
ASL	2	•	•	6	6	•	•	LDX	•	3	4	5	5	•	•
ASLD	•	•	•	•	•	3	•	LSR	2	•	•	6	6	•	•
ASR	2	•	•	6	6	•	•	LSRD	•	•	•	•	•	3	•
BCC	•	•	•	•	•	•	3	MUL	•	•	•	•	•	10	•
BCS	•	•	•	•	•	•	3	NEG	2	•	•	6	6	•	•
BEQ	•	•	•	•	•	•	3	NOP	•	•	•	•	•	2	•
BGE	•	•	•	•	•	•	3	ORA	•	2	3	4	4	•	•
BGT	•	•	•	•	•	•	3	PSH	3	•	•	•	•	•	•
BHI	•	•	•	•	•	•	3	PSHX	•	•	•	•	•	4	•
BIT	•	2	3	4	4	•	•	PUL	4	•	•	•	•	•	•
BLE	•	•	•	•	•	•	3	PULX	•	•	•	•	•	5	•
BLS	•	•	•	•	•	•	3	ROL	2	•	•	6	6	•	•
BLT	•	•	•	•	•	•	3	ROR	2	•	•	6	6	•	•
BMI	•	•	•	•	•	•	3	RTI	•	•	•	•	•	10	•
BNE	•	•	•	•	•	•	3	RTS	•	•	•	•	•	5	•
BPL	•	•	•	•	•	•	3	SBA	•	•	•	•	•	2	•
BRA	•	•	•	•	•	•	3	SBC	•	2	3	4	4	•	•
BRN	•	•	•	•	•	•	3	SEC	•	•	•	•	•	2	•
BSR	•	•	•	•	•	•	6	SEI	•	•	•	•	•	2	•
BVC	•	•	•	•	•	•	3	SEV	•	•	•	•	•	2	•
BVS	•	•	•	•	•	•	3	STA	•	•	3	4	4	•	•
CBA	•	•	•	•	•	2	•	STD	•	•	4	5	5	•	•
CLC	•	•	•	•	•	2	•	STS	•	•	4	5	5	•	•
CLI	•	•	•	•	•	2	•	STX	•	•	4	5	5	•	•
CLR	2	•	•	6	6	•	•	SUB	•	2	3	4	4	•	•
CLV	•	•	•	•	•	2	•	SUBD	•	4	5	6	6	•	•
CMP	•	2	3	4	4	•	•	SWI	•	•	•	•	•	12	•
COM	2	•	•	6	6	•	•	TAB	•	•	•	•	•	2	•
CPX	•	4	5	6	6	•	•	TAP	•	•	•	•	•	2	•
DAA	•	•	•	•	•	2	•	TBA	•	•	•	•	•	2	•
DEC	2	•	•	6	6	•	•	TPA	•	•	•	•	•	2	•
DES	•	•	•	•	•	3	•	TST	2	•	•	6	6	•	•
DEX	•	•	•	•	•	3	•	TSX	•	•	•	•	•	3	•
EOR	•	2	3	4	4	•	•	TXS	•	•	•	•	•	3	•
INC	2	•	•	6	6	•	•	WAI	•	•	•	•	•	9	•
INS	•	•	•	•	•	3	•								

- **Summary of Cycle by Cycle Operation**

Table 12 provides a detailed description of the information present on the Address Bus, Data Bus, and the Read/Write line (R/$\overline{W}$) during each cycle for each instruction.

This information is useful in comparing actual with expected results during debug of both software and hardware as the control program is executed. The information is categorized in groups according to addressing mode and number of cycles per instruction. (In general, instructions with the same addressing mode and number of cycles execute in the same manner; exceptions are indicated in the table).

Table 12 Cycle by Cycle Operation

Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/$\overline{W}$ Line	Data Bus
IMMEDIATE					
ADC EOR ADD LDA AND ORA BIT SBC CMP SUB	2	1 2	Op Code Address Op Code Address + 1	1 1	Op Code Operand Data
LDS LDX LDD	3	1 2 3	Op Code Address Op Code Address + 1 Op Code Address + 2	1 1 1	Op Code Operand Data (High Order Byte) Operand Data (Low Order Byte)
CPX SUBD ADDD	4	1 2 3 4	Op Code Address Op Code Address + 1 Op Code Address + 2 Address Bus FFFF	1 1 1 1	Op Code Operand Data (High Order Byte) Operand Data (Low Order Byte) Low Byte of Restart Vector
DIRECT					
ADC EOR ADD LDA AND ORA BIT SBC CMP SUB	3	1 2 3	Op Code Address Op Code Address + 1 Address of Operand	1 1 1	Op Code Address of Operand Operand Data
STA	3	1 2 3	Op Code Address Op Code Address + 1 Destination Address	1 1 0	Op Code Destination Address Data from Accumulator
LDS LDX LDD	4	1 2 3 4	Op Code Address Op Code Address + 1 Address of Operand Operand Address + 1	1 1 1 1	Op Code Address of Operand Operand Data (High Order Byte) Operand Data (Low Order Byte)
STS STX STD	4	1 2 3 4	Op Code Address Op Code Address + 1 Address of Operand Address of Operand + 1	1 1 0 0	Op Code Address of Operand Register Data (High Order Byte) Register Data (Low Crder Byte)
CPX SUBD ADDD	5	1 2 3 4 5	Op Code Address Op Code Address + 1 Operand Address Operand Address + 1 Address Bus FFFF	1 1 1 1 1	Op Code Address of Operand Operand Data (High Order Byte) Operand Data (Low Order Byte) Low Byte of Restart Vector
JSR	5	1 2 3 4 5	Op Code Address Op Code Address + 1 Subroutine Address Stack Pointer Stack Pointer + 1	1 1 1 0 0	Op Code Irrelevant Data First Subroutine Op Code Return Address (Low Order Byte) Return Address (High Order Byte)

(Continued)

Table 12 Cycle by Cycle Operation (Continued)

Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/$\overline{W}$ Line	Data Bus
INDEXED					
JMP	3	1	Op Code Address	1	Op Code
		2	Op Code Address + 1	1	Offset
		3	Address Bus FFFF	1	Low Byte of Restart Vector
ADC EOR	4	1	Op Code Address	1	Op Code
ADD LDA		2	Op Code Address + 1	1	Offset
AND ORA		3	Address Bus FFFF	1	Low Byte of Restart Vector
BIT SBC		4	Index Register Plus Offset	1	Operand Data
CMP SUB					
STA	4	1	Op Code Address	1	Op Code
		2	Op Code Address + 1	1	Offset
		3	Address Bus FFFF	1	Low Byte of Restart Vector
		4	Index Register Plus Offset	0	Operand Data
LDS	5	1	Op Code Address	1	Op Code
LDX		2	Op Code Address + 1	1	Offset
LDD		3	Address Bus FFFF	1	Low Byte of Restart Vector
LDD		4	Index Register Plus Offset	1	Operand Data (High Order Byte)
		5	Index Register Plus Offset + 1	1	Operand Data (Low Order Byte)
STS	5	1	Op Code Address	1	Op Code
STX		2	Op Code Address + 1	1	Offset
STD		3	Address Bus FFFF	1	Low Byte of Restart Vector
		4	Index Register Plus Offset	0	Operand Data (High Order Byte)
		5	Index Register Plus Offset + 1	0	Operand Data (Low Order Byte)
ASL LSR	6	1	Op Code Address	1	Op Code
ASR NEG		2	Op Code Address + 1	1	Offset
CLR ROL		3	Address Bus FFFF	1	Low Byte of Restart Vector
COM ROR		4	Index Register Plus Offset	1	Current Operand Data
DEC TST*		5	Address Bus FFFF	1	Low Byte of Restart Vector
INC		6	Index Register Plus Offset	0	New Operand Data
CPX	6	1	Op Code Address	1	Op Code
SUBD		2	Op Code Address + 1	1	Offset
ADDD		3	Address Bus FFFF	1	Low Byte of Restart Vector
		4	Index Register + Offset	1	Operand Data (High Order Byte)
		5	Index Register + Offset + 1	1	Operand Data (Low Order Byte)
		6	Address Bus FFFF	1	Low Byte of Restart Vector
JSR	6	1	Op Code Address	1	Op Code
		2	Op Code Address + 1	1	Offset
		3	Address Bus FFFF	1	Low Byte of Restart Vector
		4	Index Register + Offset	1	First Subroutine Op Code
		5	Stack Pointer	0	Return Address (Low Order Byte)
		6	Stack Pointer − 1	0	Return Address (High Order Byte)

* In the TST instruction, R/$\overline{W}$ line of the sixth cycle is "1" level, and AB = FFFF, DB = Low Byte of Reset Vector.

(Continued)

Table 12 Cycle by Cycle Operation (Continued)

Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/$\overline{W}$ Line	Data Bus
EXTENDED					
JMP	3	1	Op Code Address	1	Op Code
		2	Op Code Address + 1	1	Jump Address (High Order Byte)
		3	Op Code Address + 2	1	Jump Address (Low Order Byte)
ADC EOR	4	1	Op Code Address	1	Op Code
ADD LDA		2	Op Code Address + 1	1	Address of Operand (High Order Byte)
AND ORA		3	Op Code Address + 2	1	Address of Operand (Low Order Byte)
BIT SBC		4	Address of Operand	1	Operand Data
CMP SUB					
STA	4	1	Op Code Address	1	Op Code
		2	Op Code Address + 1	1	Destination Address (High Order Byte)
		3	Op Code Address + 2	1	Destination Address (Low Order Byte)
		4	Operand Destination Address	0	Data from Accumulator
LDS	5	1	Op Code Address	1	Op Code
LDX		2	Op Code Address + 1	1	Address of Operand (High Order Byte)
LDD		3	Op Code Address + 2	1	Address of Operand (Low Order Byte)
		4	Address of Operand	1	Operand Data (High Order Byte)
		5	Address of Operand + 1	1	Operand Data (Low Order Byte)
STS	5	1	Op Code Address	1	Op Code
STX		2	Op Code Address + 1	1	Address of Operand (High Order Byte)
STD		3	Op Code Address + 2	1	Address of Operand (Low Order Byte)
		4	Address of Operand	0	Operand Data (High Order Byte)
		5	Address of Operand + 1	0	Operand Data (Low Order Byte)
ASL LSR	6	1	Op Code Address	1	Op Code
ASR NEG		2	Op Code Address + 1	1	Address of Operand (High Order Byte)
CLR ROL		3	Op Code Address + 2	1	Address of Operand (Low Order Byte)
COM ROR		4	Address of Operand	1	Current Operand Data
DEC TST*		5	Address Bus FFFF	1	Low Byte of Restart Vector
INC		6	Address of Operand	0	New Operand Data
CPX	6	1	Op Code Address	1	Op Code
SUBD		2	Op Code Address + 1	1	Operand Address (High Order Byte)
ADDD		3	Op Code Address + 2	1	Operand Address (Low Order Byte)
		4	Operand Address	1	Operand Data (High Order Byte)
		5	Operand Address + 1	1	Operand Data (Low Order Byte)
		6	Address Bus FFFF	1	Low Byte of Restart Vector
JSR	6	1	Op Code Address	1	Op Code
		2	Op Code Address + 1	1	Address of Subroutine (High Order Byte)
		3	Op Code Address + 2	1	Address of Subroutine (Low Order Byte)
		4	Subroutine Starting Address	1	Op Code of Next Instruction
		5	Stack Pointer	0	Return Address (Low Order Byte)
		6	Stack Pointer – 1	0	Return Address (High Order Byte)

* In the TST instruction, R/$\overline{W}$ line of the sixth cycle is "1" level, and AB = FFFF, DB = Low Byte of Reset Vector.

(Continued)

Table 12 Cycle by Cycle Operation (Continued)

Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/$\overline{W}$ Line	Data Bus
IMPLIED					
ABA DAA SEC ASL DEC SEI ASR INC SEV CBA LSR TAB CLC NEG TAP CLI NOP TBA CLR ROL TPA CLV ROR TST COM SBA	2	1 2	Op Code Address Op Code Address + 1	1 1	Op Code Op Code of Next Instruction
ABX	3	1 2 3	Op Code Address Op Code Address + 1 Address Bus FFFF	1 1 1	Op Code Irrelevant Data Low Byte of Restart Vector
ASLD LSRD	3	1 2 3	Op Code Address Op Code Address + 1 Address Bus FFFF	1 1 1	Op Code Irrelevant Data Low Byte of Restart Vector
DES INS	3	1 2 3	Op Code Address Op Code Address + 1 Previous Register Contents	1 1 1	Op Code Op Code of Next Instruction Irrelevant Data
INX DEX	3	1 2 3	Op Code Address Op Code Address + 1 Address Bus FFFF	1 1 1	Op Code Op Code of Next Instruction Low Byte of Restart Vector
PSHA PSHB	3	1 2 3	Op Code Address Op Code Address + 1 Stack Pointer	1 1 0	Op Code Op Code of Next Instruction Accumulator Data
TSX	3	1 2 3	Op Code Address Op Code Address + 1 Stack Pointer	1 1 1	Op Code Op Code of Next Instruction Irrelevant Data
TXS	3	1 2 3	Op Code Address Op Code Address + 1 Address Bus FFFF	1 1 1	Op Code Op Code of Next Instruction Low Byte of Restart Vector
PULA PULB	4	1 2 3 4	Op Code Address Op Code Address + 1 Stack Pointer Stack Pointer + 1	1 1 1 1	Op Code Op Code of Next Instruction Irrelevant Data Operand Data from Stack
PSHX	4	1 2 3 4	Op Code Address Op Code Address + 1 Stack Pointer Stack Pointer – 1	1 1 0 0	Op Code Irrelevant Data Index Register (Low Order Byte) Index Register (High Order Byte)
PULX	5	1 2 3 4 5	Op Code Address Op Code Address + 1 Stack Pointer Stack Pointer + 1 Stack Pointer +2	1 1 1 1 1	Op Code Irrelevant Data Irrelevant Data Index Register (High Order Byte) Index Register (Low Order Byte)
RTS	5	1 2 3 4 5	Op Code Address Op Code Address + 1 Stack Pointer Stack Pointer + 1 Stack Pointer + 2	1 1 1 1 1	Op Code Irrelevant Data Irrelevant Data Address of Next Instruction (High Order Byte) Address of Next Instruction (Low Order Byte)
WAI**	9	1 2 3 4	Op Code Address Op Code Address + 1 Stack Pointer Stack Pointer – 1	1 1 0 0	Op Code Op Code of Next Instruction Return Address (Low Order Byte) Return Address (High Order Byte)

(Continued)

Table 12 Cycle by Cycle Operation (Continued)

Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/$\overline{W}$ Line	Data Bus
WAI**		5	Stack Pointer – 2	0	Index Register (Low Order Byte)
		6	Stack Pointer – 3	0	Index Register (High Order Byte)
		7	Stack Pointer – 4	0	Contents of Accumulator A
		8	Stack Pointer – 5	0	Contents of Accumulator B
		9	Stack Pointer – 6	0	Contents of Cond. Code Register
MUL	10	1	Op Code Address	1	Op Code
		2	Op Code Address + 1	1	Irrelevant Data
		3	Address Bus FFFF	1	Low Byte of Restart Vector
		4	Address Bus FFFF	1	Low Byte of Restart Vector
		5	Address Bus FFFF	1	Low Byte of Restart Vector
		6	Address Bus FFFF	1	Low Byte of Restart Vector
		7	Address Bus FFFF	1	Low Byte of Restart Vector
		8	Address Bus FFFF	1	Low Byte of Restart Vector
		9	Address Bus FFFF	1	Low Byte of Restart Vector
		10	Address Bus FFFF	1	Low Byte of Restart Vector
$\overline{RTI}$	10	1	Op Code Address	1	Op Code
		2	Op Code Address + 1	1	Irrelevant Data
		3	Stack Pointer	1	Irrelevant Data
		4	Stack Pointer + 1	1	Contents of Cond. Code Reg. from Stack
		5	Stack Pointer + 2	1	Contents of Accumulator B from Stack
		6	Stack Pointer + 3	1	Contents of Accumulator A from Stack
		7	Stack Pointer + 4	1	Index Register from Stack (High Order Byte)
		8	Stack Pointer + 5	1	Index Register from Stack (Low Order Byte)
		9	Stack Pointer + 6	1	Next Instruction Address from Stack (High Order Byte)
		10	Stack Pointer + 7	1	Next Instruction Address from Stack (Low Order Byte)
SWI	12	1	Op Code Address	1	Op Code
		2	Op Code Address + 1	1	Irrelevant Data
		3	Stack Pointer	0	Return Address (Low Order Byte)
		4	Stack Pointer – 1	0	Return Address (High Order Byte)
		5	Stack Pointer – 2	0	Index Register (Low Order Byte)
		6	Stack Pointer – 3	0	Index Register (High Order Byte)
		7	Stack Pointer – 4	0	Contents of Accumulator A
		8	Stack Pointer – 5	0	Contents of Accumulator B
		9	Stack Pointer – 6	0	Contents of Cond. Code Register
		10	Stack Pointer – 7	1	Irrelevant Data
		11	Vector Address FFFA (Hex)	1	Address of Subroutine (High Order Byte)
		12	Vector Address FFFB (Hex)	1	Address of Subroutine (Low Order Byte)

(Continued)

** While the MPU is in the "Wait" state, its bus state will appear as a series of MPU reads of an address which is seven locations less than the original contents of the Stack Pointer. Contrary to the HD6800, none of the ports are driven to the high impedance state by a WAI instruction.

Table 12 Cycle by Cycle Operation (Continued)

RELATIVE

Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/$\overline{W}$ Line	Data Bus
BCC BHT BNE BCS BLE BPL BEQ BLS BRA BGE BLT BVC BGT BMT BVS BRN	3	1	Op Code Address	1	Op Code
		2	Op Code Address + 1	1	Branch Offset
		3	Address Bus FFFF	1	Low Byte of Restart Vector
BSR	6	1	Op Code Address	1	Op Code
		2	Op Code Address + 1	1	Branch Offset
		3	Address Bus FFFF	1	Low Byte of Restart Vector
		4	Subroutine Starting Address	1	Op Code of Next Instruction
		5	Stack Pointer	0	Return Address (Low Order Byte)
		6	Stack Pointer − 1	0	Return Address (High Order Byte)

- **Summary of Undefined Instruction Operations**

The HD6803 has 36 underfined instructions. When these are carried out, the contents of Register and Memory in MPU change at random.

When the op codes (4E, 5E) are used to execute. the MPU continues to increase the program counter and it will not stop until the Reset signal enters. These op codes are used to test the LSI.

Table 13 Op codes Map

HD6803 MICROPROCESSOR INSTRUCTIONS

OP CODE HI	LO	0000	0001	0010	0011	ACC A	ACC B	IND	EXT	ACCA or SP IMM	ACCA or SP DIR	ACCA or SP IND	ACCA or SP EXT	ACCB or X IMM	ACCB or X DIR	ACCB or X IND	ACCB or X EXT	
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0000	0		SBA	BRA	TSX	NEG				SUB								0
0001	1	NOP	CBA	BRN	INS					CMP								1
0010	2			BHI	PULA (+1)					SBC								2
0011	3			BLS	PULB (+1)	COM				•	SUBD (+2)			•	ADDD (+2)			3
0100	4	LSRD (+1)		BCC	DES	LSR				AND								4
0101	5	ASLD (+1)		BCS	TXS					BIT								5
0110	6	TAP	TAB	BNE	PSHA	ROR				LDA								6
0111	7	TPA	TBA	BEQ	PSHB	ASR					STA				STA			7
1000	8	INX (+1)		BVC	PULX (+2)	ASL				EOR								8
1001	9	DEX (+1)	DAA	BVS	RTS (+2)	ROL				ADC								9
1010	A	CLV		BPL	ABX	DEC				ORA								A
1011	B	SEV	ABA	BMI	RTI (+7)					ADD								B
1100	C	CLC		BGE	PSHX (+1)	INC				•	CPX (+2)			•	LDD (+1)			C
1101	D	SEC		BLT	MUL (+7)	TST				BSR (+4)	JSR (+2)			• (+1)	STD (+1)			D
1110	E	CLI		BGT	WAI (+6)	••		JMP (−3)		•	LDS (+1)			•	LDX (+1)			E
1111	F	SEI		BLE	SWI (+9)	CLR				• (+1)	STS (+1)			• (+1)	STX (+1)			F
BYTE/CYCLE		1/2	1/2	2/3	1/3	1/2	1/2	2/6	3/6	2/2	2/3	2/4	3/4	2/2	2/3	2/4	3/4	

[NOTES] 1) Undefined Op codes are marked with [/].

2) ( ) indicate that the number in parenthesis must be added to the cycle count for that instruction.

3) The instructions shown below are all 3 bytes and are marked with "•".
Immediate addressing mode of SUBD, CPX, LDS, ADDD, LDD and LDX instructions, and undefined op codes (8F, CD, CF).

4) The Op codes (4E, 5E) are 1 byte/∞ cycles instructions, and are marked with "••".

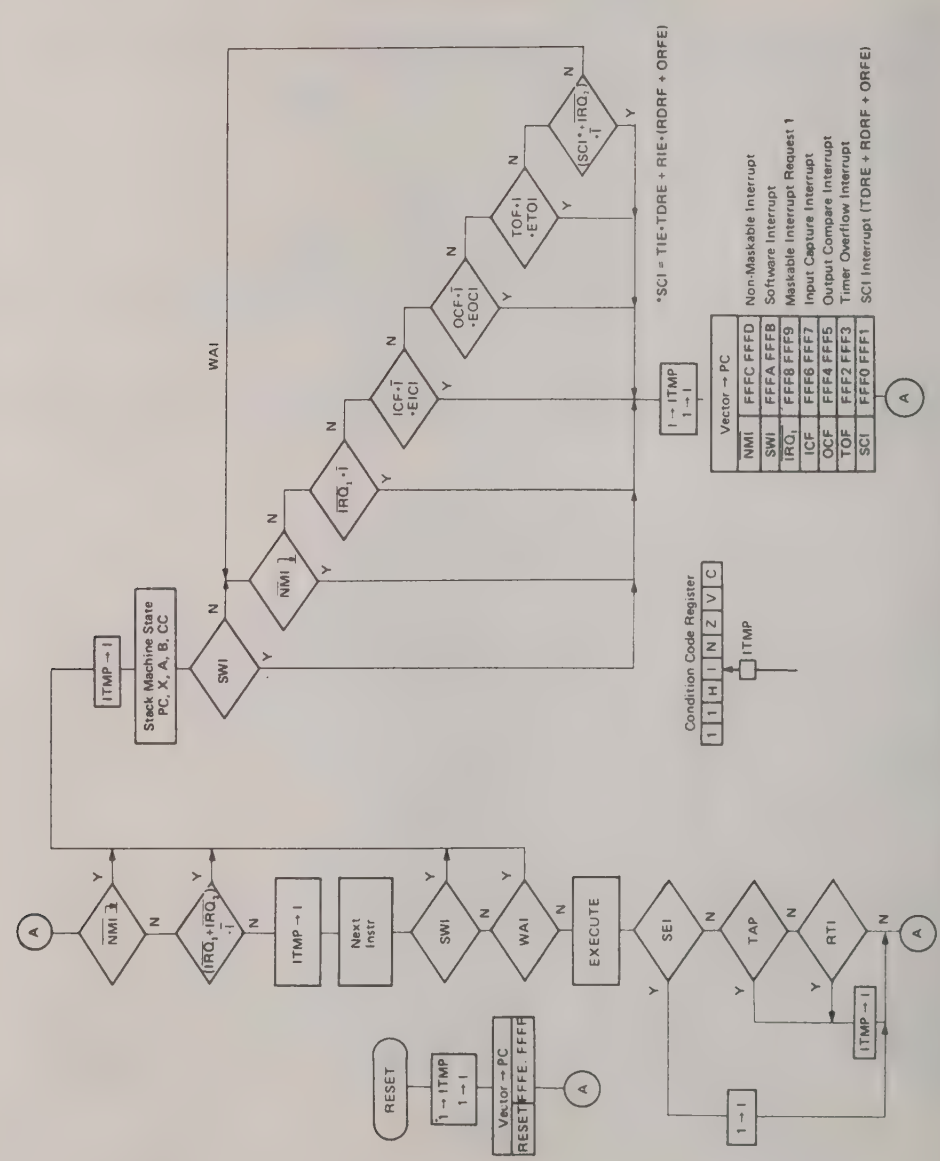


Figure 16 Interrupt Flowchart

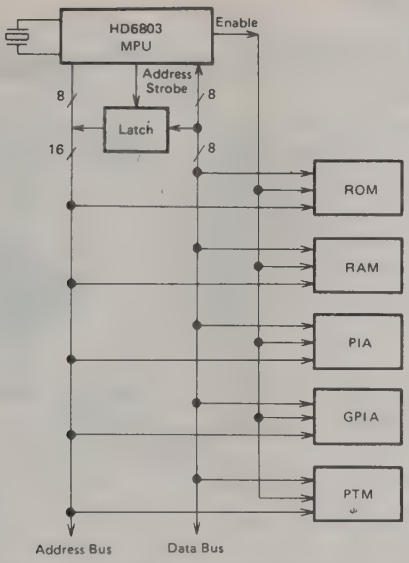


Figure 17 HD6803 MPU Expanded Multiplexed Bus

■ **Caution for the HD6803 Family SCI, TIMER Status Flag**

The flags shown in Table 14 are cleared by reading/writing (flag reset condition 2) the data register corresponding to each flag after reading the status register (flag reset condition 1).

To clear the flag correctly, take the following procedure:

1. Read the status register.
2. Test the flag.
3. Read the data register.

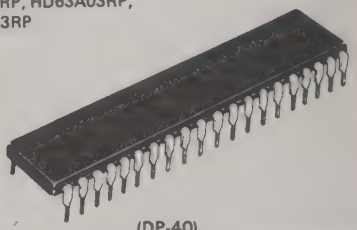
Table 14 Status Flag Reset Conditions

	Status Flag	Flag Reset Condition 1 (Status Register)	Flag Reset Condition 2 (Data Register)
TIMER	ICF	When each flag is "1", TRCSR/Read	ICR/Read
	OCF		OCR/Write
	TOF		TC/Read
SCI	RDRF	When each flag is "1", TRCSR/Read	RDR/Read
	ORFE		
	TDRE		TDR/Write

HD6303R, HD63A03R, HD63B03R
CMOS MPU (Micro Processing Unit)

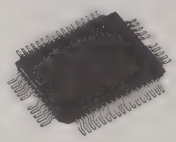
The HD6303R is an 8-bit CMOS micro processing unit which has the completely compatible instruction set with the HD6301V1. 128 bytes RAM, Serial Communication Interface (SCI), parallel I/O ports and multi function timer are incorporated in the HD6303R. It is bus compatible with HMCS6800 and can be expanded up to 65k bytes. Like the HMCS6800 family, I/O level is TTL compatible with +5.0V single power supply. As the HD6303R is CMOS MPU, power dissipation is extremely low. And also HD6303R has Sleep Mode and Stand-by Mode as lower power dissipation mode. Therefore, flexible low power consumption application is possible.

HD6303RP, HD63A03RP, HD63B03RP



(DP-40)

HD6303RF, HD63A03RF, HD63B03RF



(FP-54)

HD6303RCG, HD63A03RCG, HD63B03RCG

(CG-40)

■ FEATURES

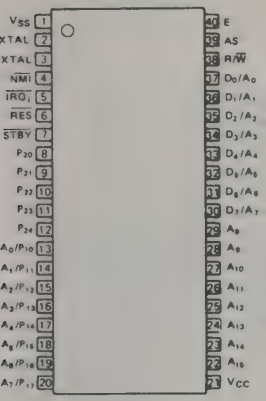
- Object Code Upward Compatible with the HD6800, HD6801, HD6802
- Multiplexed Bus ($D_0/A_0 \sim D_7/A_7 A_8 \sim A_{15}$), Non Multiplexed Bus ($D_0 \sim D_7$, $A_0 \sim A_{15}$)
- Abundant On-Chip Functions Compatible with the HD6301V1; 128 Bytes RAM, 13 Parallel I/O Lines, 16-bit Timer, Serial Communication Interface (SCI)
- Low Power Consumption Mode; Sleep Mode, Stand-By Mode
- Minimum Instruction Execution Time
  1μs (f=1MHz), 0.67μs (f=1.5MHz), 0.5μs (f=2.0MHz)
- Bit Manipulation, Bit Test Instruction
- Error Detecting Function; Address Trap, Op Code Trap
- Up to 65k Bytes Address Space
- Wide Operation Range
  V_{CC} = 3 to 6V (f = 0.1 ~ 0.5 MHz)
  f = 0.1 to 2.0 MHz (V_{CC} = 5V ± 10%)

■ TYPE OF PRODUCTS

Type No.	Bus Timing
HD6303R	1.0 MHz
HD63A03R	1.5 MHz
HD63B03R	2.0 MHz

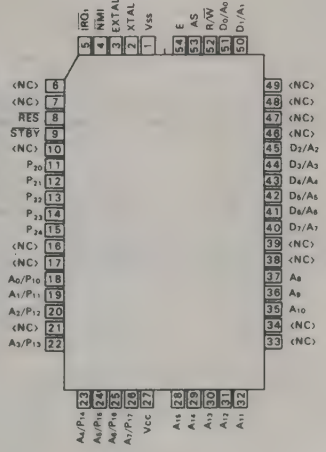
■ PIN ARRANGEMENT

- HD6303RP, HD63A03RP, HD63B03RP



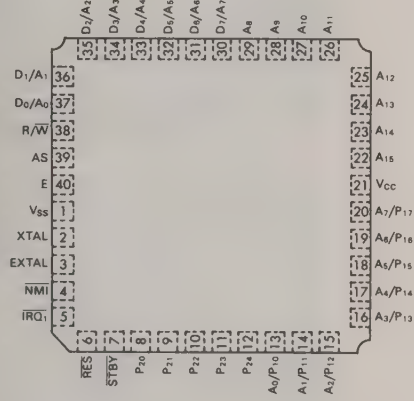
(Top View)

- HD6303RF, HD63A03RF, HD63B03RF



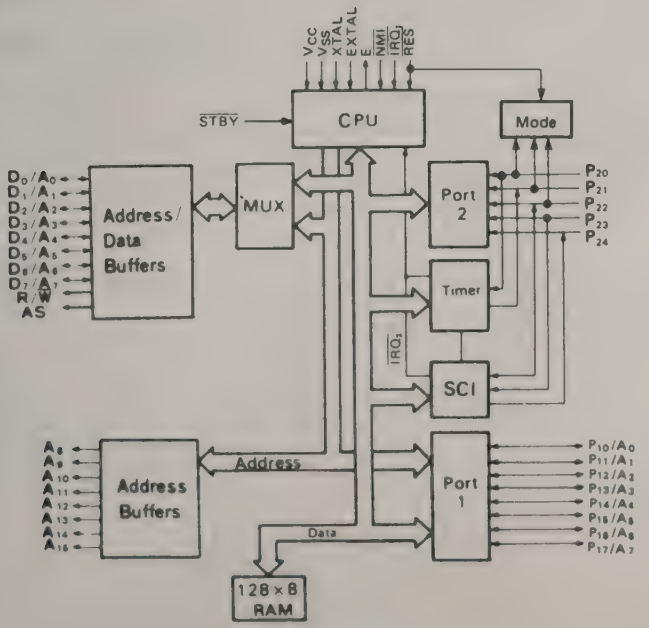
(Top View)

- HD6303RCG, HD63A03RCG, HD63B03RCG



(Top View)

■ BLOCK DIAGRAM



■ ABSOLUTE MAXIMUM RATINGS

Item	Symbol	Value	Unit
Supply Voltage	V_{CC}	-0.3 ~ +7.0	V
Input Voltage	V_{in}	-0.3 ~ V_{CC}+0.3	V
Operating Temperature	T_{opr}	0 ~ +70	°C
Storage Temperature	T_{stg}	-55 ~ +150	°C

(NOTE) This product has protection circuits in input terminal from high static electricity voltage and high electric field. But be careful not to apply overvoltage more than maximum ratings to these high input impedance protection circuits. To assure the normal operation, we recommend V_{in}, V_{out} : $V_{SS} \leqq (V_{in}$ or $V_{out}) \leqq V_{CC}$.

■ ELECTRICAL CHARACTERISTICS

● DC CHARACTERISTICS (V_{CC} = 5.0V±10%, V_{SS} = 0V, Ta = 0~+70°C, unless otherwise noted.)

Item		Symbol	Test Condition	min	typ	max	Unit
Input "High" Voltage	$\overline{RES}$, $\overline{STBY}$	V_{IH}		V_{CC}-0.5	–	V_{CC} +0.3	V
	EXTAL			V_{CC}x0.7	–		
	Other Inputs			2.0	–		
Input "Low" Voltage	All Inputs	V_{IL}		-0.3	–	0.8	V
Input Leakage Current	$\overline{NMI}$, $\overline{IRQ_1}$, $\overline{RES}$, $\overline{STBY}$	$\|I_{in}\|$	V_{in} = 0.5~V_{CC}-0.5V	–	–	1.0	μA
Three State (off-state) Leakage Current	P_{10}~P_{17}, P_{20}~P_{24}, D_0~D_7, A_8~A_{15}	$\|I_{TSI}\|$	V_{in} = 0.5~V_{CC}-0.5V	–	–	1.0	μA
Output "High" Voltage	All Outputs	V_{OH}	I_{OH} = -200μA	2.4	–	–	V
			I_{OH} = -10μA	V_{CC}-0.7	–	–	V
Output "Low" Voltage	All Outputs	V_{OL}	I_{OL} = 1.6mA	–	–	0.55	V
Input Capacitance	All Inputs	C_{in}	V_{in} = 0V, f = 1.0MHz, Ta = 25°C	–	–	12.5	pF
Standby Current	Non Operation	I_{CC}		–	2.0	15.0	μA
Current Dissipation*		I_{CC}	Operating (f=1MHz**)	–	6.0	10.0	mA
			Sleeping (f=1MHz**)	–	1.0	2.0	
RAM Stand-By Voltage		V_{RAM}		2.0	–	–	V

* V_{IH} min = V_{CC}-1.0V, V_{IL} max = 0.8V

** Current Dissipation of the operating or sleeping condition is proportional to the operating frequency. So the typ. or max. values about Current Dissipations at f = x MHz operation are decided according to the following formula;

typ. value (f = xMHz) = typ. value (f = 1MHz) x x
max. value (f = xMHz) = max. value (f = 1MHz) x x
(both the sleeping and operating)

- **AC CHARACTERISTICS** (V_{CC} = 5.0V±10%, V_{SS} = 0V, Ta = 0~+70°C, unless otherwise noted.)

BUS TIMING

Item		Symbol	Test Condition	HD6303R			HD63A03R			HD63B03R			Unit
				min	typ	max	min	typ	max	min	typ	max	
Cycle Time		t_{cyc}		1	–	10	0.666	–	10	0.5	–	10	μs
Address Strobe Pulse Width * "High"		PW_{ASH}		220	–	–	150	–	–	110	–	–	ns
Address Strobe Rise Time		t_{ASr}		–	–	20	–	–	20	–	–	20	ns
Address Strobe Fall Time		t_{ASf}		–	–	20	–	–	20	–	–	20	ns
Address Strobe Delay Time *		t_{ASD}		60	–	–	40	–	–	20	–	–	ns
Enable Rise Time		t_{Er}		–	–	20	–	–	20	–	–	20	ns
Enable Fall Time		t_{Ef}		–	–	20	–	–	20	–	–	20	ns
Enable Pulse Width "High" Level*		PW_{EH}		450	–	–	300	–	–	220	–	–	ns
Enable Pulse Width "Low" Level*		PW_{EL}		450	–	–	300	–	–	220	–	–	ns
Address Strobe to Enable Delay* Time		t_{ASED}		60	–	–	40	–	–	20	–	–	ns
Address Delay Time		t_{AD1}		–	–	250	–	–	190	–	–	160	ns
		t_{AD2}	Fig. 1	–	–	250	–	–	190	–	–	160	ns
Address Delay Time for Latch*		t_{ADL}	Fig. 2	–	–	250	–	–	190	–	–	160	ns
Data Set-up Time	Write	t_{DSW}		230	–	–	150	–	–	100	–	–	ns
	Read	t_{DSR}		80	–	–	60	–	–	50	–	–	ns
Data Hold Time	Read	t_{HR}		0	–	–	0	–	–	0	–	–	ns
	Write	t_{HW}		20	–	–	20	–	–	20	–	–	ns
Address Set-up Time for Latch*		t_{ASL}		60	–	–	40	–	–	20	–	–	ns
Address Hold Time for Latch		t_{AHL}		30	–	–	20	–	–	20	–	–	ns
Address Hold Time		t_{AH}		20	–	–	20	–	–	20	–	–	ns
A_0 ~ A_7 Set-up Time Before E*		t_{ASM}		200	–	–	110	–	–	60	–	–	ns
Peripheral Read Access Time	Non-Multiplexed Bus *	(t_{ACCN})		–	–	650	–	–	395	–	–	270	ns
	Multiplexed Bus*	(t_{ACCM})		–	–	650	–	–	395	–	–	270	ns
Oscillator stabilization Time		t_{RC}	Fig. 8	20	–	–	20	–	–	20	–	–	ms
Processor Control Set-up Time		t_{PCS}	Fig. 9	200	–	–	200	–	–	200	–	–	ns

*These timings change in approximate proportion to t_{cyc}. The figures in this characteristics represent those when t_{cyc} is minimum (= in the highest speed operation).

PERIPHERAL PORT TIMING

Item		Symbol	Test Condition	HD6303R			HD63A03R			HD63B03R			Unit
				min	typ	max	min	typ	max	min	typ	max	
Peripheral Data Set-up Time	Port 1, 2	t_{PDSU}	Fig. 3	200	–	–	200	–	–	200	–	–	ns
Peripheral Data Hold Time	Port 1, 2	t_{PDH}	Fig. 3	200	–	–	200	–	–	200	–	–	ns
Delay Time, Enable Negative Transition to Peripheral Data Valid	Port 1, 2*	t_{PWD}	Fig. 4	–	–	300	–	–	300	–	–	300	ns

* Except P_{21}

TIMER, SCI TIMING

Item	Symbol	Test Condition	HD6303R			HD63A03R			HD63B03R			Unit
			min	typ	max	min	typ	max	min	typ	max	
Timer Input Pulse Width	t_{PWT}		2.0	–	–	2.0	–	–	2.0	–	–	t_{cyc}
Delay Time, Enable Positive Transition to Timer Out	t_{TOD}	Fig. 5	–	–	400	–	–	400	–	–	400	ns
SCI Input Clock Cycle	t_{Scyc}		2.0	–	–	2.0	–	–	2.0	–	–	t_{cyc}
SCI Input Clock Pulse Width	t_{PWSCK}		0.4	–	0.6	0.4	–	0.6	0.4	–	0.6	t_{Scyc}

MODE PROGRAMMING

Item	Symbol	Test Condition	HD6303R			HD63A03R			HD63B03R			Unit
			min	typ	max	min	typ	max	min	typ	max	
$\overline{RES}$ "Low" Pulse Width	PW_{RSTL}	Fig. 6	3	–	–	3	–	–	3	–	–	t_{cyc}
Mode Programming Set-up Time	t_{MPS}		2	–	–	2	–	–	2	–	–	t_{cyc}
Mode Programming Hold Time	t_{MPH}		150	–	–	150	–	–	150	–	–	ns

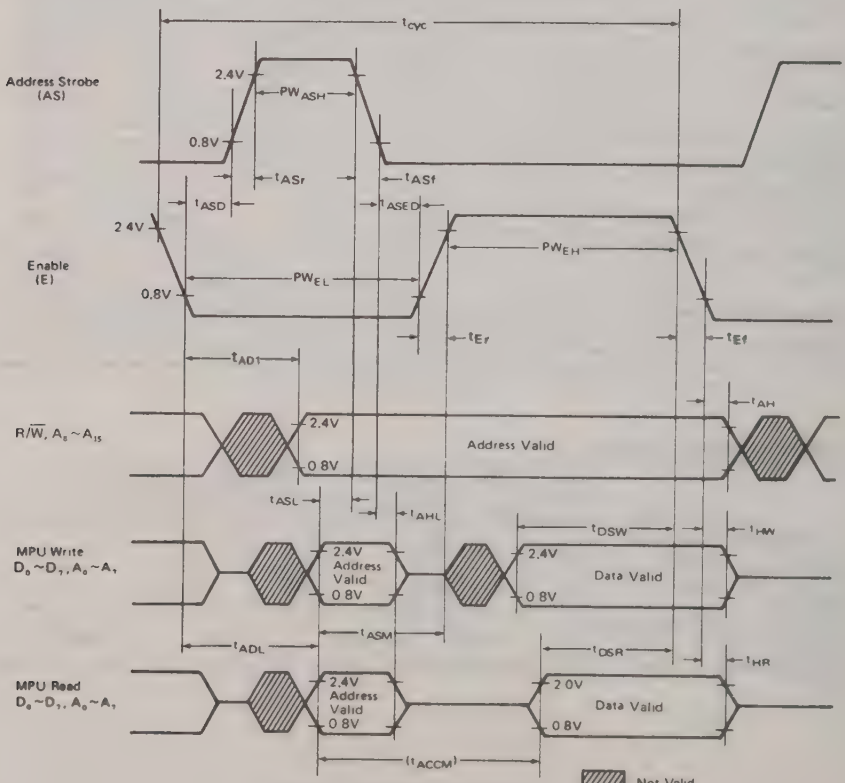


Figure 1 Multiplexed Bus Timing

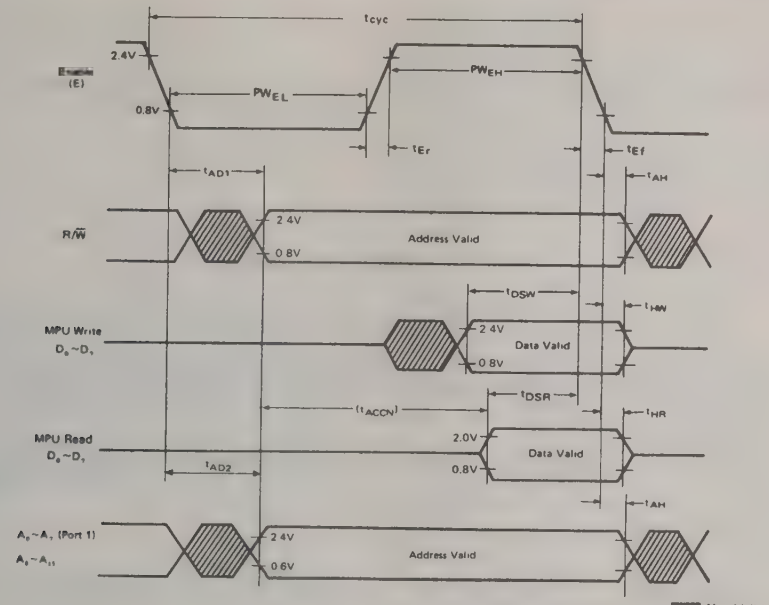


Figure 2 Non-Multiplexed Bus Timing

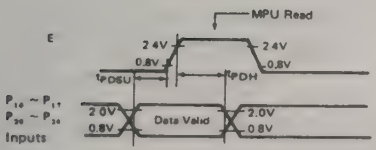
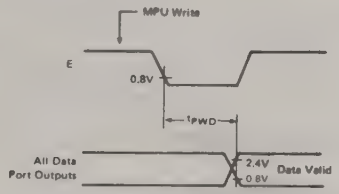


Figure 3 Port Data Set-up and Hold Times (MPU Read)



Note) Port 2: Except P_{21}

Figure 4 Port Data Delay Times (MPU Write)

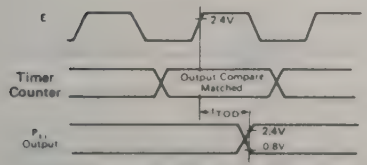


Figure 5 Timer Output Timing

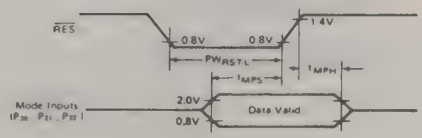


Figure 6 Mode Programming Timing

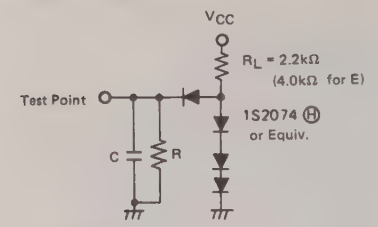


Figure 7 Bus Timing Test Loads (TTL Load)

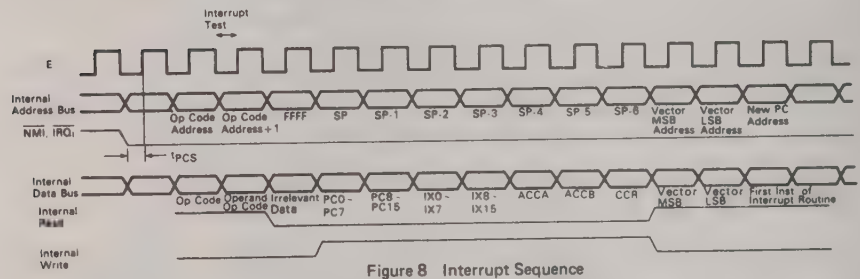


Figure 8 Interrupt Sequence

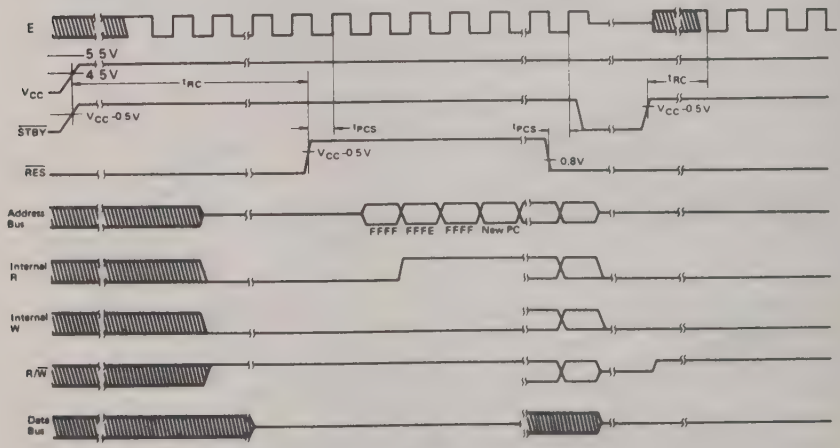


Figure 9 Reset Timing

■ FUNCTIONAL PIN DESCRIPTION

• V_{CC}, V_{SS}

These two pins are used for power supply and GND. Recommended power supply voltage is 5V ± 10%. 3 to 6V can be used for low speed operation (100 ~ 500 kHz).

• XTAL, EXTAL

These two pins are connected with parallel resonant fundamental crystal, AT cut. For instance, in order to obtain the system clock 1MHz, a 4MHz resonant fundamental crystal is used because the devide-by-4 circuitry is included. An example of the crystal interface is shown in Fig. 10. EXTAL accepts an external clock input of duty 45% to 55% to drive. For external clock, XTAL pin should be open. The crystal and capacitors should be mounted as close as possible to the pins.

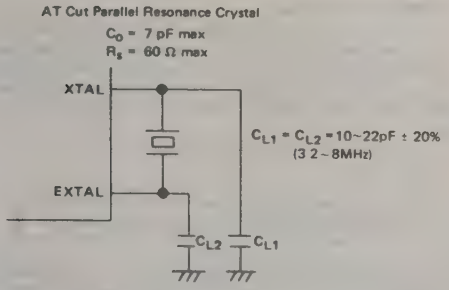


Figure 10 Crystal Interface

- **Standby ($\overline{STBY}$)**

This pin is used to place the MPU in the standby mode. If this goes to "Low" level, the oscillation stops, the internal clock is tied to V_{SS} or V_{CC} and the MPU is reset. In order to retain information in RAM during standby, write "0" into RAM enable bit (RAME). RAME is bit 6 of the RAM Control Register at address $0014. This disables the RAM, so the contents of RAM is guaranteed. For details of the standby mode, see the Standby section.

- **Reset ($\overline{RES}$)**

This input is used to reset the MPU. $\overline{RES}$ must be held "Low" for at least 20ms when the power starts up. It should be noted that, before clock generator stabilize, the internal state and I/O ports are uncertain, because MPU can not be reset without clock. To reset the MPU during system operation, it must be held "Low" for at least 3 system clock cycles. From the third cycle, all address buses become "high-impedance" and it continues while $\overline{RES}$ is "Low". If $\overline{RES}$ goes to "High", CPU does the following.

(1) I/O Port 2 bits 2,1,0 are latched into bits PC2, PC1, PC0 of program control register.
(2) The contents of the two Start Addresses, $FFFE, $FFFF are brought to the program counter, from which program starts (see Table 1).
(3) The interrupt mask bit is set. In order to have the CPU recognize the maskable interrupts $\overline{IRQ_1}$ and $\overline{IRQ_2}$, clear it before those are used.

- **Enable (E)**

This output pin supplies system clock. Output is a single-phase, TTL compatible and 1/4 the crystal oscillation frequency. It will drive two LS TTL load and 40pF capacitance.

- **Non Maskable Interrupt ($\overline{NMI}$)**

When the falling edge of the input signal of this pin is recognized, NMI sequence starts. The current instruction is continued to complete, even if $\overline{NMI}$ signal is detected. Interrupt mask bit in Condition Code Register has no effect on NMI detection. In response to NMI interrupt, the information of Program Counter, Index Register, Accumulators, and Condition Code Register are stored on the stack. On completion of this sequence, vectoring address $FFFC and $FFFD are generated to load the contents to the program counter. Then the CPU branch to a non maskable interrupt service routine.

- **Interrupt Request ($\overline{IRQ_1}$)**

This level sensitive input requests a maskable interrupt sequence. When $\overline{IRQ_1}$ goes to "Low", the CPU waits until it completes the current instruction that is being executed. Then, if the interrupt mask bit in Condition Code Register is not set, CPU begins interrupt sequence; otherwise, interrupt request is neglected.

Once the sequence has started, the information of Program Counter, Index Register, Accumulator, Condition Code Register are stored on the stack. Then the CPU sets the interrupt mask bit so that no further maskable interrupts may be responded.

Table 1 Interrupt Vectoring memory map

	Vector		Interrupt
	MSB	LSB	
Highest Priority	FFFE	FFFF	$\overline{RES}$
	FFEE	FFEF	TRAP
	FFFC	FFFD	$\overline{NMI}$
	FFFA	FFFB	Software Interrupt (SWI)
	FFF8	FFF9	$\overline{IRQ_1}$ (or IS3)
	FFF6	FFF7	ICF (Timer Input Capture)
	FFF4	FFF5	OCF (Timer Output Compare)
	FFF2	FFF3	TOF (Timer Overflow)
Lowest Priority	FFF0	FFF1	SCI (RDRF + ORFE + TDRE)

At the end of the cycle, the CPU generates 16 bit vectoring addresses indicating memory addresses $FFF8 and $FFF9, and loads the contents to the Program Counter, then branch to an interrupt service routine.

The Internal Interrupt will generate signal ($\overline{IRQ_2}$) which is quite the same as $\overline{IRQ_1}$ except that it will use the vector address $FFF0 to $FFF7.

When $\overline{IRQ_1}$ and $\overline{IRQ_2}$ are generated at the same time, the former precedes the latter. Interrupt Mask Bit in the condition code register, if being set, will keep the both interrupts off.

On occurrence of Address error or Op-code error, TRAP interrupt is invoked. This interrupt has priority next to $\overline{RES}$. Regardless of the interrupt Mask Bit condition, the CPU will start an interrupt sequence. The vector for this interrupt will be $FFEE, $FFEF.

- **Read/Write (R/$\overline{W}$)**

This TTL compatible output signals peripheral and memory devices whether CPU is in Read ("High"), or in Write ("Low"). The normal stand-by state is Read ("High"). Its output will drive one TTL load and 90pF capacitance.

- **Address Strobe (AS)**

In the multiplexed mode, address strobe signal appears at this pin. It is used to latch the lower 8 bits addresses multiplexed with data at $D_0/A_0 \sim D_7/A_7$. The 8-bit latch is controlled by address strobe as shown in Figure 15. Thereby, $D_0/A_0 \sim D_7/A_7$ can become data bus during E pulse. The timing chart of this signal is shown in Figure 1.

Address Strobe (AS) is sent out even if the internal address is accessed.

- **PORTS**

There are two I/O ports on HD6303R MPU (one 8-bit ports and one 5-bit port). Each port has an independent write-only data direction register to program individual I/O pins for input or output.*

When the bit of associated Data Direction Register is "1", I/O pin is programmed for output, if "0", then programmed for

an input.

There are two ports : Port 1, Port 2. Addresses of each port and associated Data Direction Register are shown in Table 2.

- Only one exception is bit 1 of Port 2 which becomes either a data input or a timer output. It cannot be used as an output port.

Table 2 Port and Data Direction Register Addresses

Ports	Port Address	Data Direction Register Address
I/O Port 1	$0002	$0000
I/O Port 2	$0003	$0001

- **I/O Port 1**

This is an 8-bit port, each bit being defined individually as input or outputs by associated Data Direction Register. The 8-bit output buffers have three-state capability, maintaining in high impedance state when they are used for input. In order to be read accurately, the voltage on the input lines must be more than 2.0V for logic "1" and less than 0.8V for logic "0".

These are TTL compatible. After the MPU has been reset, all I/O lines are configured as inputs in Multiplexed mode. In Non Multiplexed mode, Port 1 will be output line for lower order address lines ($A_0 \sim A_7$), which can drive one TTL load and 30 pF capacitance.

- **I/O Port 2**

This port has five lines, whose I/O direction depends on its data direction register. The 5-bit output buffers have three-state capability, going high impedance state when used as inputs. In order to be read accurately, the voltage on the input lines must be more than 2.0V for logic "1" and less than 0.8V for logic "0". After the MPU has been reset, I/O lines are configured as inputs. These pins on Port 2 ($P_{20} \sim P_{22}$ of the chip) are used to program the mode of operation during reset. The values of these three pins during reset are latched into the upper 3 bits (bit 7, 6 and 5) of Port 2 Data Register which is explained in the MODE SELECTION section.

In all modes, Port 2 can be configured as I/O lines. This port also provides access to the Serial I/O and the Timer. However, note that bit 1 (P_{21}) is the only pin restricted to data input or Timer output.

- **BUS**
- **$D_0/A_0 \sim D_7/A_7$**

This TTL compatible three-state buffer can drive one TTL load and 90 pF capacitance.

Non Multiplexed Mode

In this mode, these pins become only data bus ($D_0 \sim D_7$).

Multiplexed Mode

These pins becomes both the data bus ($D_0 \sim D_7$) and lower bits of the address bus ($A_0 \sim A_7$). An address strobe output is "High" when the address is on the pins.

- **$A_8 \sim A_{15}$**

Each line is TTL compatible and can drive one TTL load and 90 pF capacitance. After reset, these pins become output for upper order address lines ($A_8 \sim A_{15}$).

- **MODE SELECTION**

The operation mode after the reset must be determined by the user wiring the P_{20}, P_{21}, and P_{22} externally. These three pins are lower order bits; I/O 0, I/O 1, I/O 2 of Port 2. They are latched into the control bits PC0, PC1, PC2 of I/O Port 2 register when $\overline{RES}$ goes "High". I/O Port 2 Register is shown below.

Port 2 DATA REGISTER

	7	6	5	4	3	2	1	0
$0003	PC2	PC1	PC0	I/O 4	I/O 3	I/O 2	I/O 1	I/O 0

An example of external hardware used for Mode Selection is shown in Figure 11. The HD14053B is used to separate the peripheral device from the MPU during reset. It is necessary if the data may conflict between peripheral device and Mode generation circuit.

No mode can be changed through software because the bits 5, 6, and 7 of Port 2 Data Register are read-only. The mode selection of the HD6303R is shown in Table 3.

The HD6303R operates in two basic modes: (1) Multiplexed Mode, (2) Non Multiplexed Mode.

- **Multiplexed Mode**

The data bus and the lower order address bus are multiplexed in the $D_0/A_0 \sim D_7/A_7$ and can be separated by the Address Strobe.

Port 2 is configured for 5 parallel I/O or Serial I/O, or Timer, or any combination thereof. Port 1 is configured for 8 parallel I/O.

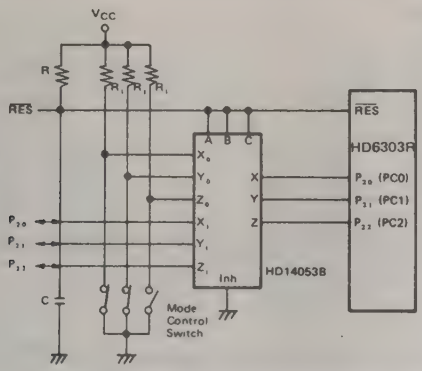
- **Non Multiplexed Mode**

In this mode, the HD6303R can directly address HMCS6800 peripherals with no address latch. $D_0/A_0 \sim D_7/A_7$ become a data bus and Port 1 becomes $A_0 \sim A_7$ address bus.

In this mode, the HD6303R is expandable up to 65k bytes with no address latch.

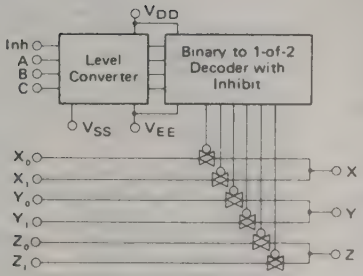
- **Lower Order Address Bus Latch**

Because the data bus is multiplexed with the lower order address bus in $D_0/A_0 \sim D_7/A_7$ in the multiplexed mode, address bits must be latched. It requires the 74LS373 Transparent octal D-type to latch the LSB. Latch connection of the HD6303R is shown in Figure 15.



Note 1) Figure of Multiplexed Mode
2) RC≈Reset Constant
3) R_1 =10kΩ

Figure 11 Recommended Circuit for Mode Selection



Truth Table

Control Input				On Switch		
Inhibit	Select			HD140538		
	C	B	A			
0	0	0	0	Z_0	Y_0	X_0
0	0	0	1	Z_0	Y_0	X_1
0	0	1	0	Z_0	Y_1	X_0
0	0	1	1	Z_0	Y_1	X_1
0	1	0	0	Z_1	Y_0	X_0
0	1	0	1	Z_1	Y_0	X_1
0	1	1	0	Z_1	Y_1	X_0
0	1	1	1	Z_1	Y_1	X_1
1	X	X	X	–		

Figure 12 HD14053B Multiplexers/De-Multiplexers

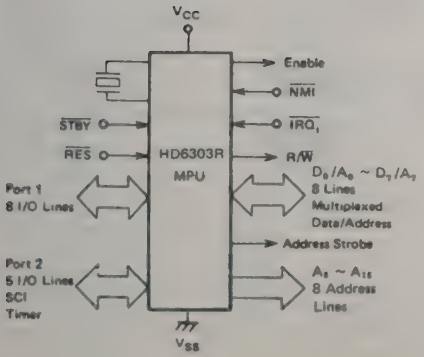


Figure 13 HD6303R MPU Multiplexed Mode

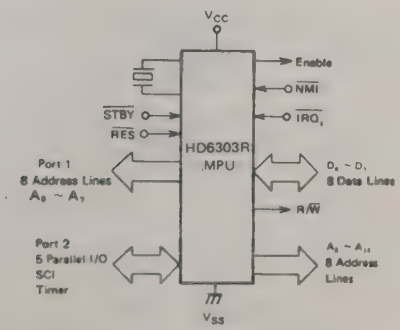
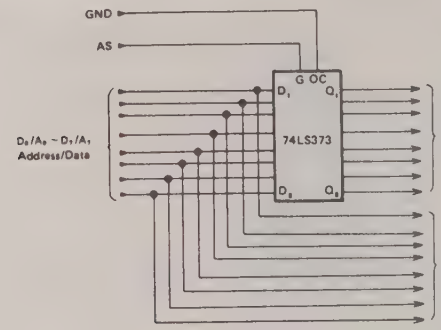


Figure 14 HD6303R MPU Non Multiplexed Mode



Function Table

Output Control (OC)	Enable G	Enable D	Output Q
L	H	H	H
L	H	L	L
L	L	X	Q_n
H	X	X	Z

Figure 15 Latch Connection

Table 3 Mode Selection

Operating Mode	P_{20}	P_{21}	P_{22}
Multiplexed Mode	L	H	L
	L	L	H
Non Multiplexed Mode	H	L	L

L: logic "0"
H: logic "1"

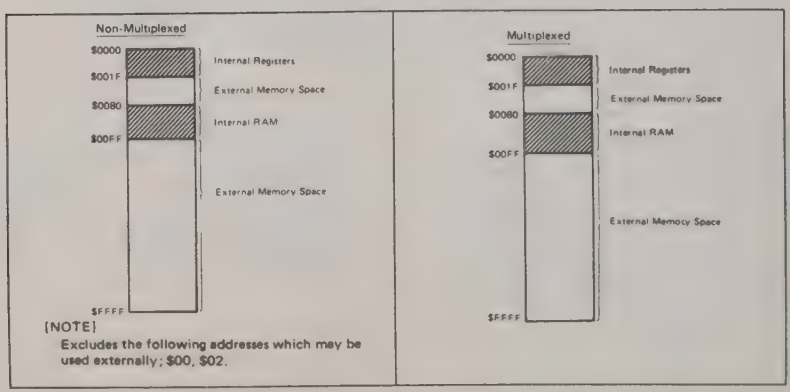
■ MEMORY MAP

The MPU can provide up to 65k byte address space. Figure 16 shows a memory map for each operating mode. The first 32 locations of each map are for the CPU's internal register only, as shown in Table 4.

Table 4 Internal Register Area

Register	Address
Port 1 Data Direction Register**	00*
Port 2 Data Direction Register**	01
Port 1 Data Register	02*
Port 2 Data Register	03
Timer Control and Status Register	08
Counter (High Byte)	09
Counter (Low Byte)	0A
Output Compare Register (High Byte)	0B
Output Compare Register (Low Byte)	0C
Input Capture Register (High Byte)	0D
Input Capture Register (Low Byte)	0E
Rate and Mode Control Register	10
Transmit/Receive Control and Status Register	11
Receive Data Register	12
Transmit Data Register	13
RAM Control Register	14
Reserved	15-1F

* External address in Non Multiplexed Mode
** 1 = Output, 0 = Input



[NOTE]
Excludes the following addresses which may be used externally; $00, $02.

Figure 16 HD6303R Memory Maps

■ PROGRAMMABLE TIMER

The HD6303R contains 16-bit programmable timer which may measure input waveform. In addition to that it can generate an output waveform by itself. For both input and output waveform, the pulse width may vary from a few microseconds to several seconds.

The timer hardware consists of

- an 8-bit control and status register
- a 16-bit free running counter
- a 16-bit output compare register
- a 16-bit input capture register

A block diagram of the timer is shown in Figure 17.

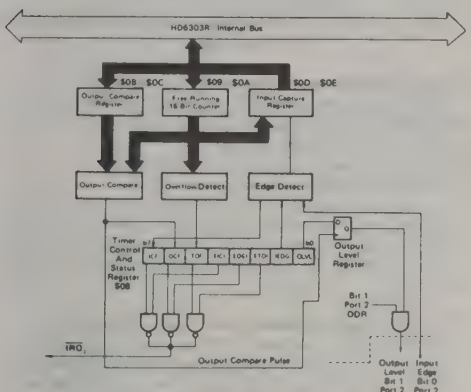


Figure 17 Programmable Timer Block Diagram

• Free Running Counter ($0009: $000A)

The key element in the programmable timer is a 16-bit free running counter, that is driven by an E (Enable) clock to increment its values. The counter value will be read out by the CPU software at any time with no effects on the counter. Reset will clear the counter.

When the MSB of this counter is read, the LSB is stored in temporary latch. The data is fetched from this latch by the subsequent read of LSB. Thus consistent double byte data can be read from the counter.

When the CPU writes arbitrary data to the MSB ($09), the value of $FFF8 is being pre-set to the counter ($09, $0A) regardless of the write data value. Then the CPU writes arbitrary data to the LSB ($0A), the data is set to the "Low" byte of the counter, at the same time, the data preceedingly written in the MSB ($09) is set to "High" byte of the counter.

When the data is written to this counter, a double byte store instruction (ex. STD) must be used. If only the MSB of counter is written, the counter is set to $FFF8.

The counter value written to the counter using the double byte store instruction is shown in Figure 18.

To write to the counter can disturb serial operations, so it should be inhibited during using the SCI. If external clock mode is used for SCI, this will not disturb serial operations.

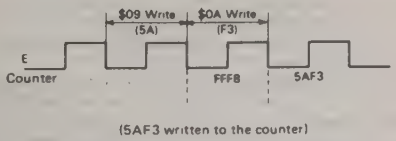


Figure 18 Counter Write Timing

• Output Compare Register ($000B:$000C)

This is a 16-bit read/write register which is used to control an output waveform. The contents of this register are constantly being compared with current value of the free running counter.

When the contents match with the value of the free running counter, a flag (OCF) in the timer control/status register (TCSR) is set and the current value of an output level Bit (OLVL) in the TCSR is transferred to Port 2 bit 1. When bit 1 of the Port 2 data direction register is "1" (output), the OLVL value will appear on the bit 1 of Port 2. Then, the value of Output Compare Register and Output level bit may be changed for the next compare.

The output compare register is set to $FFFF during reset.

The compare function is inhibited at the cycle of writing to the high byte of the output compare register and at the cycle just after that to ensure valid compare. It is also inhibited in same manner at writing to the free running counter.

In order to write a data to Output Compare Register, a double byte store instruction (ex.STD) must be used.

• Input Capture Register ($000D: $000E)

The input capture register is a 16-bit read-only register used to hold the current value of free running counter captured when the proper transition of an external input signal occurs.

The input transition change required to trigger the counter transfer is controlled by the input edge bit (IEDG).

To allow the external input signal to go in the edge detect unit, the bit of the Data Direction Register corresponding to bit 0 of Port 2 must have been cleared (to zero).

To insure input capture in all cases, the width of an input pulse requires at least 2 Enable cycles.

• Timer Control/Status Register (TCSR) ($0008)

This is an 8-bit register. All 8-bits are readable and the lower 5 bits may be written. The upper 3 bits are read-only, indicating the timer status information as is shown below.

(1) A proper transition has been detected on the input pin (ICF).
(2) A match has been found between the value in the free running counter and the output compare register (OCF).
(3) When counting up to $0000 (TOF).

Each flag has an individual enable bit in TCSR which determines whether or not an interrupt request may occur ($\overline{IRQ_2}$). If the I-bit in Condition Code Register has been cleared, a prior vectored address occurs corresponding to each flag. A description of each bit is as follows.

Timer Control / Status Register

7	6	5	4	3	2	1	0	
ICF	OCF	TOF	EICI	EOCI	ETOI	IEDG	OLVL	$0008

Bit 0 OLVL (Output Level); When a match is found in the value between the counter and the output com-

pare register, this bit is transferred to the Port 2 bit 1. If the DDR corresponding to Port 2 bit 1 is set "1", the value will appear on the output pin of Port 2 bit 1.

Bit 1 IEDG (Input Edge): This bit control which transition of an input of Port 2 bit 0 will trigger the data transfer from the counter to the input capture register. The DDR corresponding to Port 2 bit 0 must be clear in advance of using this function.
When IEDG = 0, trigger takes place on a negative edge ("High"-to-"Low" transition). When IEDG = 1, trigger takes place on a positive edge ("Low"-to-"High" transition).

Bit 2 ETOI (Enable Timer Overflow Interrupt); When set, this bit enables TOF interrupt to generate the interrupt request ($\overline{IRQ_2}$). When cleared, the interrupt is inhibited.

Bit 3 EOCI (Enable Output Compare Interrupt); When set, this bit enables OCF interrupt to generate the interrupt request ($\overline{IRQ_2}$). When cleared, the interrupt is inhibited.

Bit 4 EICI (Enable Input Capture Interrupt); When set, this bit enables ICF interrupt to generate the interrupt request ($\overline{IRQ_2}$). When cleared, the interrupt is inhibited.

Bit 5 TOF (Timer Over Flow Flag); This read-only bit is set at the transition of $FFFF to $0000 of the counter. It is cleared by CPU read of TCSR (with TOF set) followed by a CPU read of the counter ($0009).

Bit 6 OCF (Output Compare Flag); This read-only bit is set when a match is found in the value between the output compare register and the counter. It is cleared by a read of TCSR (with OCF set) followed by a CPU write to the output compare register ($000B or $000C).

Bit 7 ICF (Input Capture Flag); The read-only bit is set by a proper transition on the input, and is cleared by a read of TCSR (with ICF set) followed by a CPU read of Input Capture Register ($000D).

Reset will clear each bit of Timer Control and Status Register.

■ SERIAL COMMUNICATION INTERFACE

The HD6303R contains a full-duplex asynchronous Serial Communication Interface (SCI). SCI may select the several kinds of the data rate. It consists of a transmitter and a receiver which operate independently but with the same data format and the same data rate. Both of transmitter and receiver communicate with the CPU via the data bus and with the outside world through Port 2 bit 2, 3 and 4. Description of hardware, software and register is as follows.

● Wake-Up Feature

In typical multiprocessor applications the software protocol will usually have the designated address at the initial byte of the message. The purpose of Wake-Up feature is to have the non-selected MPU neglect the remainder of the message. Thus the non-selected MPU can inhibit the all further interrupt process until the next message begins.

Wake-Up feature is re-enabled by a ten consecutive "1"s which indicates an idle transmit line. Therefore software protocol must put an idle period between the messages and must prevent it within the message.

With this hardware feature, the non-selected MPU is re-enabled (or "waked-up") by the next message.

● Programmable Options

The HD6303R has the following programmable features.

- data format; standard mark/space (NRZ)
- clock source; external or internal
- baud rate; one of 4 rates per given E clock frequency or 1/8 of external clock
- wake-up feature; enabled or disabled
- interrupt requests; enabled or masked individually for transmitter and receiver
- clock output; internal clock enabled or disabled to Port 2 bit 2
- Port 2 (bits 3, 4); dedicated or not dedicated to serial I/O individually

● Serial Communication Hardware

The serial communications hardware is controlled by 4 registers as shown in Figure 19. The registers include:

- an 8-bit control/status register
- a 4-bit rate/mode control register (write-only)
- an 8-bit read-only receive data register
- an 8-bit write-only transmit data register

Besides these 4 registers, Serial I/O utilizes Port 2 bit 3 (input) and bit 4 (output). Port 2 bit 2 can be used when an option is selected for the internal-clock-out or the external-clock-in.

● Transmit/Receive Control Status Register (TRCSR)

TRCS Register consists of 8 bits which all may be read while only bits 0 to 4 may be written. The register is initialized to $20 on $\overline{RES}$. The bits of the TRCS Register are explained below.

Transmit / Receive Control Status Register

7	6	5	4	3	2	1	0	
RDRF	ORFE	TDRE	RIE	RE	TIE	TE	WU	ADDR $0011

Bit 0 WU (Wake Up); Set by software and cleared by hardware on receipt of ten consecutive "1"s. While this bit is "1", RDRF and ORFE flags are not set even if data are received or errors are detected. Therefore received data are ignored. It should be noted that RE flag must have already been set in advance of WU flag's set.

Bit 1 TE (Transmit Enable); This bit enables transmitter. When this bit is set, bit 4 of Port 2 DDR is also forced to be set. It remains set even if TE is cleared. Preamble of ten consecutive "1"s is transmitted just after this bit is set, and then transmitter becomes ready to send data. If this bit is cleared, the transmitter is disabled and serial I/O affects nothing on Port 2 bit 4.

Bit 2 TIE (Transmit Interrupt Enable); When this bit is set, TDRE (bit 5) causes an $\overline{IRQ_2}$ interrupt. When cleared, TDRE interrupt is masked.

Bit 3 RE (Receive Enable); When set, Port 2 bit 3 can be used as an input of receive regardless of DDR value for this bit. When cleared, the receiver is disabled.

Bit 4 RIE (Receive Interrupt Enable); When this bit is set, RDRF (bit 7) or ORFE (bit 6) cause an $\overline{IRQ_2}$ interrupt. When cleared, this interrupt is masked.

Bit 5 TDRE (Transmit Data Register Empty); When the data is transferred from the Transmit Data Register to Output Shift Register, this bit is set by hardware. The bit is cleared by reading the status register followed by writing the next new data into the Transmit Data Register. TDRE is initialized to 1 by $\overline{RES}$.

Bit 6 ORFE (Over Run Framing Error); When overrun or framing error occurs (receive only), this bit is set by hardware. Over Run Error occurs if the attempt is made to transfer the new byte to the receive data register while the RDRF is "1". Framing Error occurs when the bit counter is not synchronized with the boundary of the byte in the receiving bit stream. When Framing Error is detected, RDRF is not set. Therefore Framing Error can be distinguished from Overrun Error. That is, if ORFE is "1" and RDRF is "1", Overrun Error is detected. Otherwise Framing Error occurs. The bit is cleared by reading the status register followed by reading the receive data register, or by $\overline{RES}$.

Bit 7 RDRF (Receive Data Register Full); This bit is set by hardware when the data is transferred from the receive shift register to the receive data register. It is cleared by reading the status register followed by reading the receive data register, or by $\overline{RES}$.

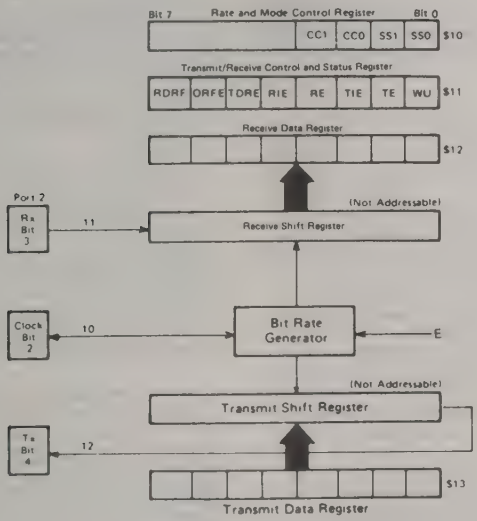


Figure 19 Serial I/O Register

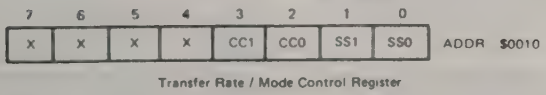


Table 5 SCI Bit Times and Transfer Rates

SS1	SS0	XTAL	2.4576 MHz	4.0 MHz	4.9152MHz
		E	614.4 kHz	1.0 MHz	1.2288MHz
0	0	E ÷ 16	26 μs/38,400 Baud	16 μs/62,500 Baud	13 μs/76,800Baud
0	1	E ÷ 128	208μs/4,800 Baud	128 μs/7812.5 Baud	104.2μs/ 9,600Baud
1	0	E ÷ 1024	1.67ms/600 Baud	1.024ms/976.6 Baud	833.3μs/ 1,200Baud
1	1	E ÷ 4096	6.67ms/150 Baud	4.096ms/244.1 Baud	3.333ms/ 300Baud

Table 6 SCI Format and Clock Source Control

CC1	CC0	Format	Clock Source	Port 2 Bit 2	Port 2 Bit 3	Port 2 Bit 4
0	0	–	–	–	–	–
0	1	NRZ	Internal	Not Used***	**	**
1	0	NRZ	Internal	Output*	**	**
1	1	NRZ	External	Input	**	**

* Clock output is available regardless of values for bits RE and TE.
** Bit 3 is used for serial input if RE = "1" in TRCS.
Bit 4 is used for serial output if TE = "1" in TRCS.
*** This pin can be used as I/O port.

- **Transfer Rate/Mode Control Register (RMCR)**

The register controls the following serial I/O functions:

- Bauds rate
- data format
- clock source
- Port 2 bit 2 feature

It is 4-bit write-only register, cleared by $\overline{RES}$. The 4 bits are considered as a pair of 2-bit fields. The lower 2 bits control the bit rate of internal clock while the upper 2 bits control the format and the clock select logic.

Bit 0 SS0
Bit 1 SS1 } Speed Select

These bits select the Baud rate for the internal clock. The rates selectable are function of E clock frequency of the CPU. Table 5 lists the available Baud Rates.

Bit2 CC0
Bit 3 CC1 } Clock Control/Format Select

They control the data format and the clock select logic. Table 6 defines the bit field.

- **Internally Generated Clock**

If the user wish to use externally an internal clock of the serial I/O, the following requirements should be noted.

- CC1, CC0 must be set to "10".
- The maximum clock rate must be E/16.
- The clock rate is equal to the bit rate.
- The values of RE and TE have no effect.

- **Externally Generated Clock**

If the user wish to supply an external clock to the Serial I/O, the following requirements should be noted.

- The CC1, CC0 must be set to "11" (See Table 6).
- The external clock must be set to 8 times of the desired baud rate.
- The maximum external clock frequency is E/2 clock.

- **Serial Operations**

The serial I/O hardware must be initialized by the software before operation. The sequence will be normally as follows.

- Writing the desired operation control bits of the Rate and Mode Control Register.
- Writing the desired operation control bits of the TRCS register.

If Port 2 bit 3, 4 are used for serial I/O, TE, RE bits may be kept set. When TE, RE bit are cleared during SCI operation, and subsequently set again, it should be noted that TE, RE must be kept "0" for at least one bit time of the current baud rate. If TE, RE are set again within one bit time, there may be the case where the initializing of internal function for transmitter and receiver does not take place correctly.

- **Transmit Operation**

Data transmission is enabled by the TE bit in the TRCS register. When set, the output of the transmit shift register is connected with Port 2 bit 4 which is unconditionally configured as an output.

After $\overline{RES}$, the user should initialize both the RMC register and the TRCS register for desired operation. Setting the TE bit causes a transmission of ten-bit preamble of "1"s. Following the preamble, internal synchronization is established and the transmitter is ready to operate. Then either of the following states exists.

(1) If the transmit data register is empty (TDRE = 1), the consecutive "1"s are transmitted indicating an idle states.
(2) If the data has been loaded into the Transmit Data Register (TDRE = 0), it is transferred to the output shift register and data transmission begins.

During the data transfer, the start bit ("0") is first transferred. Next the 8-bit data (beginning at bit 0) and finally the stop bit ("1"). When the contents of the Transmit Data Register is transferred to the output shift register, the hardware sets the TDRE flag bit: If the CPU fails to respond to the flag within the proper time, TDRE is kept set and then a continuous string of 1's is sent until the data is supplied to the data register.

- **Receive Operation**

The receive operation is enabled by the RE bit. The serial input is connected with Port 2 bit 3. The receiver operation is determined by the contents of the TRCS and RMC register. The received bit stream is synchronized by the first "0" (start bit). During 10-bit time, the data is strobed approximately at the center of each bit. If the tenth bit is not "1" (stop bit), the system assumes a framing error and the ORFE is set.

If the tenth bit is "1", the data is transferred to the receive data register, and the RDRF flag is set. If the tenth bit of the next data is received and still RDRF is preserved set, then ORFE is set indicating that an overrun error has occurred.

After the CPU read of the status register as a response to RDRF flag or ORFE flag, followed by the CPU read of the receive data register, RDRF or ORFE will be cleared.

■ RAM CONTROL REGISTER

The register assigned to the address $0014 gives a status information about standby RAM.

RAM Control Register

	7	6	5	4	3	2	1	0
$0014	STBY PWR	RAME	x	x	x	x	x	x

Bit 0 Not used.
Bit 1 Not used.
Bit 2 Not used.

Bit 3 Not used.
Bit 4 Not used.
Bit 5 Not used.
Bit 6 RAM Enable (RAME).

Using this control bit, the user can disable the RAM. RAM Enable bit is set on the positive edge of $\overline{RES}$ and RAM is enabled. The program can write "1" or "0". If RAME is cleared, the RAM address becomes external address and the CPU may read the data from the outside memory.

Bit 7 Standby Power Bit (STBY PWR)

This bit can be read or written by the user program. It is cleared when the V_{CC} voltage is removed. Normally this bit is set by the program before going into stand-by mode. When the CPU recovers from stand-by mode, this bit should be checked. If it is "1", the data of the RAM is retained during stand-by and it is valid.

■ GENERAL DESCRIPTION OF INSTRUCTION SET

The HD6303R has an upward object code compatible with the HD6801 to utilize all instruction sets of the HMCS6800. The execution time of the key instruction is reduced to increase the system through-put. In addition, the bit operation instruction, the exchange instruction between the index and the accumulator, the sleep instruction are added. This section describes:

- CPU programming model (See Fig. 20)
- Addressing modes
- Accumulator and memory manipulation instructions (See Table 7)
- New instructions
- Index register and stack manipulation instructions (See Table 8)
- Jump and branch instructions (See Table 9)
- Condition code register manipulation instructions (See Table 10)
- Op-code map (See Table 11)
- Cycle-by-cycle operation (See Table 12)

• CPU Programming Model

The programming model for the HD6303R is shown in Figure 20. The double accumulator is physically the same as the accumulator A concatenated with the accumulator B, so that the contents of A and B is changed with executing operation of an accumulator D.

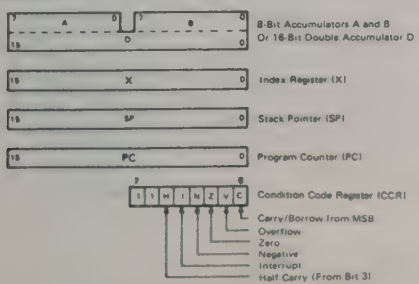


Figure 20 CPU Programming Model

• CPU Addressing Modes

The HD6303R has seven address modes which depend on both of the instruction type and the code. The address mode for every instruction is shown along with execution time given in terms of machine cycles (Table 7 to 11). When the clock frequency is 4 MHz, the machine cycle will be microseconds.

Accumulator (ACCX) Addressing

Only the accumulator (A or B) is addressed. Either accumulator A or B is specified by one-byte instructions.

Immediate Addressing

In this mode, the operand is stored in the second byte of the instruction except that the operand in LDS and LDX, etc are stored in the second and the third byte. These are two or three-byte instructions.

Direct Addressing

In this mode, the second byte of instruction indicates the address where the operand is stored. Direct addressing allows the user to directly address the lowest 256 bytes in the machine; locations zero through 255. Improved execution times are achieved by storing data in these locations. For system configuration, it is recommended that these locations should be RAM and be utilized preferably for user's data realm. These are two-byte instructions except the AIM, OIM, EIM and TIM which have three-byte.

Extended Addressing

In this mode, the second byte indicates the upper 8 bit addresses where the operand is stored, while the third byte indicates the lower 8 bits. This is an absolute address in memory. These are three-byte instructions.

Indexed Addressing

In this mode, the contents of the second byte is added to the lower 8 bits in the Index Register. For each of AIM, OIM, EIM and TIM instructions, the contents of the third byte are added to the lower 8 bits in the Index Register. In addition, the resulting "carry" is added to the upper 8 bits in the Index Register. The result is used for addressing memory. Because the modified address is held in the Temporary Address Register, there is no change to the Index Register. These are two-byte instructions but AIM, OIM, EIM, TIM have three-byte.

Implied Addressing

In this mode, the instruction itself gives the address; stack pointer, index register, etc. These are 1-byte instructions.

Relative Addressing

In this mode, the contents of the second byte is added to the lower 8 bits in the program counter. The resulting carry or borrow is added to the upper 8 bits. This helps the user to address the data within a range of −126 to +129 bytes of the current execution instruction. These are two-byte instructions.

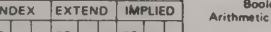
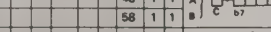
Table 7 Accumulator, Memory Manipulation Instructions

Operations	Mnemonic	IMMED			DIRECT			INDEX			EXTEND			IMPLIED			Boolean/ Arithmetic Operation	5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#		H	I	N	Z	V	C
Add	ADDA	8B	2	2	9B	3	2	AB	4	2	BB	4	3				A + M → A	↕	•	↕	↕	↕	↕
	ADDB	CB	2	2	DB	3	2	EB	4	2	FB	4	3				B + M → B	↕	•	↕	↕	↕	↕
Add Double	ADDD	C3	3	3	D3	4	2	E3	5	2	F3	5	3				A B + M M + 1 → A B	•	•	↕	↕	↕	↕
Add Accumulators	ABA													1B	1	1	A + B → A	↕	•	↕	↕	↕	↕
Add With Carry	ADCA	89	2	2	99	3	2	A9	4	2	B9	4	3				A + M + C → A	↕	•	↕	↕	↕	↕
	ADCB	C9	2	2	D9	3	2	E9	4	2	F9	4	3				B + M + C → B	↕	•	↕	↕	↕	↕
AND	ANDA	84	2	2	94	3	2	A4	4	2	B4	4	3				A·M → A	•	•	↕	↕	R	•
	ANDB	C4	2	2	D4	3	2	E4	4	2	F4	4	3				B·M → B	•	•	↕	↕	R	•
Bit Test	BIT A	85	2	2	95	3	2	A5	4	2	B5	4	3				A·M	•	•	↕	↕	R	•
	BIT B	C5	2	2	D5	3	2	E5	4	2	F5	4	3				B·M	•	•	↕	↕	R	•
Clear	CLR							6F	5	2	7F	5	3				00 → M	•	•	R	S	R	R
	CLRA													4F	1	1	00 → A	•	•	R	S	R	R
	CLRB													5F	1	1	00 → B	•	•	R	S	R	R
Compare	CMPA	81	2	2	91	3	2	A1	4	2	B1	4	3				A - M	•	•	↕	↕	↕	↕
	CMPB	C1	2	2	D1	3	2	E1	4	2	F1	4	3				B - M	•	•	↕	↕	↕	↕
Compare Accumulators	CBA													11	1	1	A - B	•	•	↕	↕	↕	↕
Complement, 1's	COM							63	6	2	73	6	3				$\overline{M}$ → M	•	•	↕	↕	R	S
	COMA													43	1	1	$\overline{A}$ → A	•	•	↕	↕	R	S
	COMB													53	1	1	$\overline{B}$ → B	•	•	↕	↕	R	S
Complement, 2's (Negate)	NEG							60	6	2	70	6	3				00 - M → M	•	•	↕	↕	①	②
	NEGA													40	1	1	00 - A → A	•	•	↕	↕	①	②
	NEGB													50	1	1	00 - B → B	•	•	↕	↕	①	②
Decimal Adjust, A	DAA													19	2	1	Converts binary add of BCD characters into BCD format	•	•	↕	↕	↕	③
Decrement	DEC							6A	6	2	7A	6	3				M - 1 → M	•	•	↕	↕	④	•
	DECA													4A	1	1	A - 1 → A	•	•	↕	↕	④	•
	DECB													5A	1	1	B - 1 → B	•	•	↕	↕	④	•
Exclusive OR	EORA	88	2	2	98	3	2	A8	4	2	B8	4	3				A ⊕ M → A	•	•	↕	↕	R	•
	EORB	C8	2	2	D8	3	2	E8	4	2	F8	4	3				B ⊕ M → B	•	•	↕	↕	R	•
Increment	INC							6C	6	2	7C	6	3				M + 1 → M	•	•	↕	↕	⑤	•
	INCA													4C	1	1	A + 1 → A	•	•	↕	↕	⑤	•
	INCB													5C	1	1	B + 1 → B	•	•	↕	↕	⑤	•
Load Accumulator	LDAA	86	2	2	96	3	2	A6	4	2	B6	4	3				M → A	•	•	↕	↕	R	•
	LDAB	C6	2	2	D6	3	2	E6	4	2	F6	4	3				M → B	•	•	↕	↕	R	•
Load Double Accumulator	LDD	CC	3	3	DC	4	2	EC	5	2	FC	5	3				M + 1 → B, M → A	•	•	↕	↕	R	•
Multiply Unsigned	MUL													3D	7	1	A x B → A : B	•	•	•	•	•	⑪
OR, Inclusive	ORAA	8A	2	2	9A	3	2	AA	4	2	BA	4	3				A + M → A	•	•	↕	↕	R	•
	ORAB	CA	2	2	DA	3	2	EA	4	2	FA	4	3				B + M → B	•	•	↕	↕	R	•
Push Data	PSHA													36	4	1	A → Msp, SP - 1 → SP	•	•	•	•	•	•
	PSHB													37	4	1	B → Msp, SP - 1 → SP	•	•	•	•	•	•
Pull Data	PULA													32	3	1	SP + 1 → SP, Msp → A	•	•	•	•	•	•
	PULB													33	3	1	SP + 1 → SP, Msp → B	•	•	•	•	•	•
Rotate Left	ROL							69	6	2	79	6	3				M / A / B — C b7 b0	•	•	↕	↕	⑥	↕
	ROLA													49	1	1		•	•	↕	↕	⑥	↕
	ROLB													59	1	1		•	•	↕	↕	⑥	↕
Rotate Right	ROR							66	6	2	76	6	3				M / A / B — C b7 b0	•	•	↕	↕	⑥	↕
	RORA													46	1	1		•	•	↕	↕	⑥	↕
	RORB													56	1	1		•	•	↕	↕	⑥	↕

Note) Condition Code Register will be explained in Note of Table 10.

(to be continued)

Table 7 Accumulator, Memory Manipulation Instructions

Operations	Mnemonic	Addressing Modes															Boolean/ Arithmetic Operation	Condition Code Register					
		IMMED			DIRECT			INDEX			EXTEND			IMPLIED				5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#		H	I	N	Z	V	C
Shift Left Arithmetic	ASL							68	6	2	78	6	3				M	•	•	↕	↕	⑥	↕
	ASLA													48	1	1	A						
	ASLB													58	1	1	B						
Double Shift Left, Arithmetic	ASLD													05	1	1		•	•	↕	↕	⑥	↕
Shift Right Arithmetic	ASR							67	6	2	77	6	3				M	•	•	↕	↕	⑥	↕
	ASRA													47	1	1	A	•	•	↕	↕	⑥	↕
	ASRB													57	1	1	B	•	•	↕	↕	⑥	↕
Shift Right Logical	LSR							64	6	2	74	6	3				M	•	•	R	↕	⑥	↕
	LSRA													44	1	1	A	•	•	R	↕	⑥	↕
	LSRB													54	1	1	B	•	•	R	↕	⑥	↕
Double Shift Right Logical	LSRD													04	1	1		•	•	R	↕	⑥	↕
Store Accumulator	STAA				97	3	2	A7	4	2	B7	4	3				A → M	•	•	↕	↕	R	•
	STAB				D7	3	2	E7	4	2	F7	4	3				B → M	•	•	↕	↕	R	•
Store Double Accumulator	STD				DD	4	2	ED	5	2	FD	5	3				A → M B → M + 1	•	•	↕	↕	R	•
Subtract	SUBA	80	2	2	90	3	2	A0	4	2	B0	4	3				A – M → A	•	•	↕	↕	↕	↕
	SUBB	C0	2	2	D0	3	2	E0	4	2	F0	4	3				B – M → B	•	•	↕	↕	↕	↕
Double Subtract	SUBD	83	3	3	93	4	2	A3	5	2	B3	5	3				A : B – M : M + 1 → A : B	•	•	↕	↕	↕	↕
Subtract Accumulators	SBA													10	1	1	A – B → A	•	•	↕	↕	↕	↕
Subtract With Carry	SBCA	82	2	2	92	3	2	A2	4	2	B2	4	3				A – M – C → A	•	•	↕	↕	↕	↕
	SBCB	C2	2	2	D2	3	2	E2	4	2	F2	4	3				B – M – C → B	•	•	↕	↕	↕	↕
Transfer Accumulators	TAB													16	1	1	A → B	•	•	↕	↕	R	•
	TBA													17	1	1	B → A	•	•	↕	↕	R	•
Test Zero or Minus	TST							6D	4	2	7D	4	3				M – 00	•	•	↕	↕	R	R
	TSTA													4D	1	1	A – 00	•	•	↕	↕	R	R
	TSTB													5D	1	1	B – 00	•	•	↕	↕	R	R
And Immediate	AIM				71	6	3	61	7	3							M·IMM→M	•	•	↕	↕	R	•
OR Immediate	OIM				72	6	3	62	7	3							M+IMM→M	•	•	↕	↕	R	•
EOR Immediate	EIM				75	6	3	65	7	3							M⊕IMM→M	•	•	↕	↕	R	•
Test Immediate	TIM				7B	4	3	6B	5	3							M·IMM	•	•	↕	↕	R	•

Note) Condition Code Register will be explained in Note of Table 10.

- **New Instructions**

In addition to the HD6801 Instruction Set, the HD6303R has the following new instructions:

AIM----(M) · (IMM) → (M)
Evaluates the AND of the immediate data and the memory, places the result in the memory.

OIM----(M) + (IMM) → (M)
Evaluates the OR of the immediate data and the memory, places the result in the memory.

EIM----(M) ⊕ (IMM) → (M)
Evaluates the EOR of the immediate data and the memory, places the result in the memory.

TIM----(M) · (IMM)
Evaluates the AND of the immediate data and the memory, changes the flag of associated condition code register

Each instruction has three bytes; the first is op-code, the second is immediate data, the third is address modifier.

XGDX--(ACCD) ↔ (IX)
Exchanges the contents of accumulator and the index register.

SLP----The MPU is brought to the sleep mode. For sleep mode, see the "sleep mode" section.

Table 8 Index Register, Stack Manipulation Instructions

Pointer Operations	Mnemonic	Addressing Modes															Boolean/ Arithmetic Operation	Condition Code Register					
		IMMED.			DIRECT			INDEX			EXTEND			IMPLIED				5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#		H	I	N	Z	V	C
Compare Index Reg	CPX	8C	3	3	9C	4	2	AC	5	2	BC	5	3				X – M M + 1	•	•	↕	↕	↕	↕
Decrement Index Reg	DEX													09	1	1	X – 1 → X	•	•	•	↕	•	•
Decrement Stack Pntr	DES													34	1	1	SP – 1 → SP	•	•	•	•	•	•
Increment Index Reg	INX													08	1	1	X + 1 → X	•	•	•	↕	•	•
Increment Stack Pntr	INS													31	1	1	SP + 1 → SP	•	•	•	•	•	•
Load Index Reg	LDX	CE	3	3	DE	4	2	EE	5	2	FE	5	3				M → X_H, (M + 1) → X_L	•	•	⑦	↕	R	•
Load Stack Pntr	LDS	8E	3	3	9E	4	2	AE	5	2	BE	5	3				M → SP_H, (M + 1) → SP_L	•	•	⑦	↕	R	•
Store Index Reg	STX				DF	4	2	EF	5	2	FF	5	3				X_H → M, X_L → (M + 1)	•	•	⑦	↕	R	•
Store Stack Pntr	STS				9F	4	2	AF	5	2	BF	5	3				SP_H → M, SP_L → (M + 1)	•	•	⑦	↕	R	•
Index Reg → Stack Pntr	TXS													35	1	1	X – 1 → SP	•	•	•	•	•	•
Stack Pntr → Index Reg	TSX													30	1	1	SP + 1 → X	•	•	•	•	•	•
Add	ABX													3A	1	1	B + X → X	•	•	•	•	•	•
Push Data	PSHX													3C	5	1	X_L → M_{SP}, SP – 1 → SP X_H → M_{SP}, SP – 1 → SP	•	•	•	•	•	•
Pull Data	PULX													38	4	1	SP + 1 → SP, M_{SP} → X_H SP + 1 → SP, M_{SP} → X_L	•	•	•	•	•	•
Exchange	XGDX													18	2	1	ACCD↔IX	•	•	•	•	•	•

Note) Condition Code Register will be explained in Note of Table 10.

Table 9 Jump, Branch Instruction

Operations	Mnemonic	Addressing Modes															Branch Test	Condition Code Register					
		RELATIVE			DIRECT			INDEX			EXTEND			IMPLIED				5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#		H	I	N	Z	V	C
Branch Always	BRA	20	3	2													None	•	•	•	•	•	•
Branch Never	BRN	21	3	2													None	•	•	•	•	•	•
Branch If Carry Clear	BCC	24	3	2													C = 0	•	•	•	•	•	•
Branch If Carry Set	BCS	25	3	2													C = 1	•	•	•	•	•	•
Branch If = Zero	BEQ	27	3	2													Z = 1	•	•	•	•	•	•
Branch If ≥ Zero	BGE	2C	3	2													N ⊕ V = 0	•	•	•	•	•	•
Branch If > Zero	BGT	2E	3	2													Z + (N ⊕ V) = 0	•	•	•	•	•	•
Branch If Higher	BHI	22	3	2													C + Z = 0	•	•	•	•	•	•
Branch If ≤ Zero	BLE	2F	3	2													Z + (N ⊕ V) = 1	•	•	•	•	•	•
Branch If Lower Or Same	BLS	23	3	2													C + Z = 1	•	•	•	•	•	•
Branch If < Zero	BLT	2D	3	2													N ⊕ V = 1	•	•	•	•	•	•
Branch If Minus	BMI	2B	3	2													N = 1	•	•	•	•	•	•
Branch If Not Equal Zero	BNE	26	3	2													Z = 0	•	•	•	•	•	•
Branch If Overflow Clear	BVC	28	3	2													V = 0	•	•	•	•	•	•
Branch If Overflow Set	BVS	29	3	2													V = 1	•	•	•	•	•	•
Branch If Plus	BPL	2A	3	2													N = 0	•	•	•	•	•	•
Branch To Subroutine	BSR	8D	5	2														•	•	•	•	•	•
Jump	JMP							6E	3	2	7E	3	3					•	•	•	•	•	•
Jump To Subroutine	JSR				9D	5	2	AD	5	2	BD	6	3					•	•	•	•	•	•
No Operation	NOP													01	1	1	Advances Prog. Cntr Only	•	•	•	•	•	•
Return From Interrupt	RTI													3B	10	1		——			⑧	——	
Return From Subroutine	RTS													39	5	1		•	•	•	•	•	•
Software Interrupt	SWI													3F	12	1		•	S	•	•	•	•
Wait for Interrupt*	WAI													3E	9	1		•	⑨	•	•	•	•
Sleep	SLP													1A	4	1		•	•	•	•	•	•

Note) *WAI puts R/$\overline{W}$ high; Address Bus goes to FFFF; Data Bus goes to the three state.
Condition Code Register will be explained in Note of Table 10.

Table 10 Condition Code Register Manipulation Instructions

Operations	Mnemonic	Addressing Modes IMPLIED OP	~	#	Boolean Operation	Condition Code Register 5 H	4 I	3 N	2 Z	1 V	0 C
Clear Carry	CLC	0C	1	1	0 → C	•	•	•	•	•	R
Clear Interrupt Mask	CLI	0E	1	1	0 → I	•	R	•	•	•	•
Clear Overflow	CLV	0A	1	1	0 → V	•	•	•	•	R	•
Set Carry	SEC	0D	1	1	1 → C	•	•	•	•	•	S
Set Interrupt Mask	SEI	0F	1	1	1 → I	•	S	•	•	•	•
Set Overflow	SEV	0B	1	1	1 → V	•	•	•	•	S	•
Accumulator A → CCR	TAP	06	1	1	A → CCR	———			⑪		———
CCR → Accumulator A	TPA	07	1	1	CCR → A	•	* •	•	•	•	•

[NOTE 1] Condition Code Register Notes: (Bit set if test is true and cleared otherwise)

① (Bit V) Test: Result = 10000000?
② (Bit C) Test: Result = 00000000?
③ (Bit C) Test: BCD Character of high-order byte greater than 9? (Not cleared if previously set)
④ (Bit V) Test: Operand = 10000000 prior to execution?
⑤ (Bit V) Test: Operand = 01111111 prior to execution?
⑥ (Bit V) Test: Set equal to N⊕C=1 after the execution of instructions
⑦ (Bit N) Test: Result less than zero? (Bit 15=1)
⑧ (All Bit) Load Condition Code Register from Stack.
⑨ (Bit I) Set when interrupt occurs. If previously set, a Non-Maskable Interrupt is required to exit the wait state.
⑩ (All Bit) Set according to the contents of Accumulator A.
⑪ (Bit C) Result of Multiplication Bit 7=1 of ACCB?

[NOTE 2] CLI instruction and interrupt.
If interrupt mask-bit is set (I="1") and interrupt is requested ($\overline{IRQ_1}$ = "0" or $\overline{IRQ_2}$ = "0"), and then CLI instruction is executed, the CPU responds as follows.

① The next instruction of CLI is one-machine cycle instruction.
Subsequent two instructions are executed before the interrupt is responded.
That is, the next and the next of the next instruction are executed.
② The next instruction of CLI is two-machine cycle (or more) instruction.
Only the next instruction is executed and then the CPU jump to the interrupt routine.
Even if TAP instruction is used, instead of CLI, the same thing occurs.

Table 11 OP-Code Map

OP CODE HI / LO	0 0000	1 0001	2 0010	3 0011	ACC A 4 0100	ACC B 5 0101	IND 6 0110	EXT / DIR 7 0111	ACCA or SP IMM 8 1000	DIR 9 1001	IND A 1010	EXT B 1011	ACCB or X IMM C 1100	DIR D 1101	IND E 1110	EXT F 1111	HI / LO
0 0000		SBA	BRA	TSX	NEG				SUB								0
1 0001	NOP	CBA	BRN	INS			AIM		CMP								1
2 0010			BHI	PULA			OIM		SBC								2
3 0011			BLS	PULB	COM				SUBD				ADDD				3
4 0100	LSRD		BCC	DES	LSR				AND								4
5 0101	ASLD		BCS	TXS			EIM		BIT								5
6 0110	TAP	TAB	BNE	PSHA	ROR				LDA								6
7 0111	TPA	TBA	BEQ	PSHB	ASR						STA				STA		7
8 1000	INX	XGDX	BVC	PULX	ASL				EOR								8
9 1001	DEX	DAA	BVS	RTS	ROL				ADC								9
A 1010	CLV	SLP	BPL	ABX	DEC				ORA								A
B 1011	SEV	ABA	BMI	RTI			TIM		ADD								B
C 1100	CLC		BGE	PSHX	INC					CPX				LDD			C
D 1101	SEC		BLT	MUL	TST				BSR		JSR				STD		D
E 1110	CLI		BGT	WAI			JMP			LDS				LDX			E
F 1111	SEI		BLE	SWI	CLR						STS				STX		F
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	

UNDEFINED OP CODE

• Only for instructions of AIM, OIM, EIM, TIM

- **Instruction Execution Cycles**

In the HMCS6800 series, the execution cycle of each instruction is the number of cycles between the start of the current instruction fetch and just before the start of the subsequent instruction fetch.

The HD6303R uses a mechanism of the pipeline control for the instruction fetch and the subsequent instruction fetch is performed during the current instruction being executed.

Therefore, the method to count instruction cycles used in the HMCS6800 series cannot be applied to the instruction cycles such as MULT, PULL, DAA and XGDX in the HD6303R.

Table 12 provides the information about the relationship among each data on the Address Bus, Data Bus, and R/$\overline{W}$ status in cycle-by-cycle basis during the execution of each instruction.

Table 12 Cycle-by-Cycle Operation

Address Mode & Instructions		Cycles	Cycle #	Address Bus	R/$\overline{W}$	Data Bus
IMMEDIATE						
ADC	ADD	2	1	Op Code Address+1	1	Operand Data
AND	BIT		2	Op Code Address+2	1	Next Op Code
CMP	EOR					
LDA	ORA					
SBC	SUB					
ADDD	CPX	3	1	Op Code Address+1	1	Operand Data (MSB)
LDD	LDS		2	Op Code Address+2	1	Operand Data (LSB)
LDX	SUBD		3	Op Code Address+3	1	Next Op Code
DIRECT						
ADC	ADD	3	1	Op Code Address+1	1	Address of Operand (LSB)
AND	BIT		2	Address of Operand	1	Operand Data
CMP	EOR		3	Op Code Address+2	1	Next Op Code
LDA	ORA					
SBC	SUB					
STA		3	1	Op Code Address+1	1	Destination Address
			2	Destination Address	0	Accumulator Data
			3	Op Code Address+2	1	Next Op Code
ADDD	CPX	4	1	Op Code Address+1	1	Address of Operand (LSB)
LDD	LDS		2	Address of Operand	1	Operand Data (MSB)
LDX	SUBD		3	Address of Operand+1	1	Operand Data (LSB)
			4	Op Code Address+2	1	Next Op Code
STD	STS	4	1	Op Code Address+1	1	Destination Address (LSB)
STX			2	Destination Address	0	Register Data (MSB)
			3	Destination Address+1	0	Register Data (LSB)
			4	Op Code Address+2	1	Next Op Code
JSR		5	1	Op Code Address+1	1	Jump Address (LSB)
			2	FFFF	1	Restart Address (LSB)
			3	Stack Pointer	0	Return Address (LSB)
			4	Stack Pointer−1	0	Return Address (MSB)
			5	Jump Address	1	First Subroutine Op Code
TIM		4	1	Op Code Address+1	1	Immediate Data
			2	Op Code Address+2	1	Address of Operand (LSB)
			3	Address of Operand	1	Operand Data
			4	Op Code Address+3	1	Next Op Code
AIM	EIM	6	1	Op Code Address+1	1	Immediate Data
OIM			2	Op Code Address+2	1	Address of Operand (LSB)
			3	Address of Operand	1	Operand Data
			4	FFFF	1	Restart Address (LSB)
			5	Address of Operand	0	New Operand Data
			6	Op Code Address+3	1	Next Op Code

— Continued —

Table 12 Cycle-by-Cycle Operation (Continued)

Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/$\overline{W}$	Data Bus
INDEXED					
JMP	3	1	Op Code Address+1	1	Offset
		2	FFFF	1	Restart Address (LSB)
		3	Jump Address	1	First Op Code of Jump Routine
ADC ADD AND BIT CMP EOR LDA ORA SBC SUB TST	4	1	Op Code Address+1	1	Offset
		2	FFFF	1	Restart Address (LSB)
		3	IX+Offset	1	Operand Data
		4	Op Code Address+2	1	Next Op Code
STA	4	1	Op Code Address+1	1	Offset
		2	FFFF	1	Restart Address (LSB)
		3	IX+Offset	0	Accumulator Data
		4	Op Code Address+2	1	Next Op Code
ADDD CPX LDD LDS LDX SUBD	5	1	Op Code Address+1	1	Offset
		2	FFFF	1	Restart Address (LSB)
		3	IX+Offset	1	Operand Data (MSB)
		4	IX+Offset+1	1	Operand Data (LSB)
		5	Op Code Address+2	1	Next Op Code
STD STS STX	5	1	Op Code Address+1	1	Offset
		2	FFFF	1	Restart Address (LSB)
		3	IX+Offset	0	Register Data (MSB)
		4	IX+Offset+1	0	Register Data (LSB)
		5	Op Code Address+2	1	Next Op Code
JSR	5	1	Op Code Address+1	1	Offset
		2	FFFF	1	Restart Address (LSB)
		3	Stack Pointer	0	Return Address (LSB)
		4	Stack Pointer−1	0	Return Address (MSB)
		5	IX+Offset	1	First Subroutine Op Code
ASL ASR COM DEC INC LSR NEG ROL ROR	6	1	Op Code Address+1	1	Offset
		2	FFFF	1	Restart Address (LSB)
		3	IX+Offset	1	Operand Data
		4	FFFF	1	Restart Address (LSB)
		5	IX+Offset	0	New Operand Data
		6	Op Code Address+2	1	Next Op Code
TIM	5	1	Op Code Address+1	1	Immediate Data
		2	Op Code Address+2	1	Offset
		3	FFFF	1	Restart Address (LSB)
		4	IX+Offset	1	Operand Data
		5	Op Code Address+3	1	Next Op Code
CLR	5	1	Op Code Address+1	1	Offset
		2	FFFF	1	Restart Address (LSB)
		3	IX+Offset	1	Operand Data
		4	IX+Offset	0	00
		5	Op Code Address+2	1	Next Op Code
AIM EIM OIM	7	1	Op Code Address+1	1	Immediate Data
		2	Op Code Address+2	1	Offset
		3	FFFF	1	Restart Address (LSB)
		4	IX+Offset	1	Operand Data
		5	FFFF	1	Restart Address (LSB)
		6	IX+Offset	0	New Operand Data
		7	Op Code Address+3	1	Next Op Code

— Continued —

Table 12 Cycle-by-Cycle Operation (Continued)

Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/$\overline{W}$	Data Bus
EXTEND					
JMP	3	1	Op Code Address+1	1	Jump Address (MSB)
		2	Op Code Address+2	1	Jump Address (LSB)
		3	Jump Address	1	Next Op Code
ADC ADD TST AND BIT CMP EOR LDA ORA SBC SUB	4	1	Op Code Address+1	1	Address of Operand (MSB)
		2	Op Code Address+2	1	Address of Operand (LSB)
		3	Address of Operand	1	Operand Data
		4	Op Code Address+3	1	Next Op Code
STA	4	1	Op Code Address+1	1	Destination Address (MSB)
		2	Op Code Address+2	1	Destination Address (LSB)
		3	Destination Address	0	Accumulator Data
		4	Op Code Address+3	1	Next Op Code
ADDD CPX LDD LDS LDX SUBD	5	1	Op Code Address+1	1	Address of Operand (MSB)
		2	Op Code Address+2	1	Address of Operand (LSB)
		3	Address of Operand	1	Operand Data (MSB)
		4	Address of Operand+1	1	Operand Data (LSB)
		5	Op Code Address+3	1	Next Op Code
STD STS STX	5	1	Op Code Address+1	1	Destination Address (MSB)
		2	Op Code Address+2	1	Destination Address (LSB)
		3	Destination Address	0	Register Data (MSB)
		4	Destination Address+1	0	Register Data (LSB)
		5	Op Code Address+3	1	Next Op Code
JSR	6	1	Op Code Address+1	1	Jump Address (MSB)
		2	Op Code Address+2	1	Jump Address (LSB)
		3	FFFF	1	Restart Address (LSB)
		4	Stack Pointer	0	Return Address (LSB)
		5	Stack Pointer−1	0	Return Address (MSB)
		6	Jump Address	1	First Subroutine Op Code
ASL ASR COM DEC INC LSR NEG ROL ROR	6	1	Op Code Address+1	1	Address of Operand (MSB)
		2	Op Code Address+2	1	Address of Operand (LSB)
		3	Address of Operand	1	Operand Data
		4	FFFF	1	Restart Address (LSB)
		5	Address of Operand	0	New Operand Data
		6	Op Code Address+3	1	Next Op Code
CLR	5	1	Op Code Address+1	1	Address of Operand (MSB)
		2	Op Code Address+2	1	Address of Operand (LSB)
		3	Address of Operand	1	Operand Data
		4	Address of Operand	0	00
		5	Op Code Address+3	1	Next Op Code

— Continued —

Table 12 Cycle-by-Cycle Operation (Continued)

Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/$\overline{W}$	Data Bus
IMPLIED					
ABA ABX ASL ASLD ASR CBA CLC CLI CLR CLV COM DEC DES DEX INC INS INX LSR LSRD ROL ROR NOP SBA SEC SEI SEV TAB TAP TBA TPA TST TSX TXS	1	1	Op Code Address+1	1	Next Op Code
DAA XGDX	2	1	Op Code Address+1	1	Next Op Code
		2	FFFF	1	Restart Address (LSB)
PULA PULB	3	1	Op Code Address+1	1	Next Op Code
		2	FFFF	1	Restart Address (LSB)
		3	Stack Pointer+1	1	Data from Stack
PSHA PSHB	4	1	Op Code Address+1	1	Next Op Code
		2	FFFF	1	Restart Address (LSB)
		3	Stack Pointer	0	Accumulator Data
		4	Op Code Address+1	1	Next Op Code
PULX	4	1	Op Code Address+1	1	Next Op Code
		2	FFFF	1	Restart Address (LSB)
		3	Stack Pointer+1	1	Data from Stack (MSB)
		4	Stack Pointer+2	1	Data from Stack (LSB)
PSHX	5	1	Op Code Address+1	1	Next Op Code
		2	FFFF	1	Restart Address (LSB)
		3	Stack Pointer	0	Index Register (LSB)
		4	Stack Pointer−1	0	Index Register (MSB)
		5	Op Code Address+1	1	Next Op Code
RTS	5	1	Op Code Address+1	1	Next Op Code
		2	FFFF	1	Restart Address (LSB)
		3	Stack Pointer+1	1	Return Address (MSB)
		4	Stack Pointer+2	1	Return Address (LSB)
		5	Return Address	1	First Op Code of Return Routine
MUL	7	1	Op Code Address+1	1	Next Op Code
		2	FFFF	1	Restart Address (LSB)
		3	FFFF	1	Restart Address (LSB)
		4	FFFF	1	Restart Address (LSB)
		5	FFFF	1	Restart Address (LSB)
		6	FFFF	1	Restart Address (LSB)
		7	FFFF	1	Restart Address (LSB)

— Continued —

Table 12 Cycle-by-Cycle Operation (Continued)

Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/$\overline{W}$	Data Bus
IMPLIED					
WAI		1	Op Code Address+1	1	Next Op Code
		2	FFFF	1	Restart Address (LSB)
		3	Stack Pointer	0	Return Address (LSB)
		4	Stack Pointer−1	0	Return Address (MSB)
	9	5	Stack Pointer−2	0	Index Register (LSB)
		6	Stack Pointer−3	0	Index Register (MSB)
		7	Stack Pointer−4	0	Accumulator A
		8	Stack Pointer−5	0	Accumulator B
		9	Stack Pointer−6	0	Conditional Code Register
RTI		1	Op Code Address+1	1	Next Op Code
		2	FFFF	1	Restart Address (LSB)
		3	Stack Pointer +1	1	Conditional Code Register
		4	Stack Pointer+2	1	Accumulator B
	10	5	Stack Pointer+3	1	Accumulator A
		6	Stack Pointer+4	1	Index Register (MSB)
		7	Stack Pointer+5	1	Index Register (LSB)
		8	Stack Pointer+6	1	Return Address (MSB)
		9	Stack Pointer+7	1	Return Address (LSB)
		10	Return Address	1	First Op Code of Return Routine
SWI		1	Op Code Address+1	1	Next Op Code
		2	FFFF	1	Restart Address (LSB)
		3	Stack Pointer	0	Return Address (LSB)
		4	Stack Pointer − 1	0	Return Address (MSB)
		5	Stack Pointer − 2	0	Index Register (LSB)
	12	6	Stack Pointer − 3	0	Index Register (MSB)
		7	Stack Pointer − 4	0	Accumulator A
		8	Stack Pointer − 5	0	Accumulator B
		9	Stack Pointer − 6	0	Conditional Code Register
		10	Vector Address FFFA	1	Address of SWI Routine (MSB)
		11	Vector Address FFFB	1	Address of SWI Routine (LSB)
		12	Address of SWI Routine	1	First Op Code of SWI Routine
SLP		1	Op Code Address+1	1	Next Op Code
		2	FFFF	1	Restart Address (LSB)
			FFFF		High Impedance-Non MPX Mode
	4	Sleep			Address Bus -MPX Mode
		3	FFFF		Restart Address (LSB)
		4	Op Code Address+1		Next Op Code

— Continued —

Table 12 Cycle-by-Cycle Operation (Continued)

Address Mode & Instructions		Cycles	Cycle #	Address Bus	R/$\overline{W}$	Data Bus
RELATIVE						
BCC	BCS	3	1	Op Code Address+1	1	Branch Offset
BEQ	BGE		2	FFFF	1	Restart Address (LSB)
BGT	BHI		3	Branch Address……Test="1" Op Code Address+1…Test="0"	1	First Op Code of Branch Routine Next Op Code
BLE	BLS					
BLT	BMT					
BNE	BPL					
BRA	BRN					
BVC	BVS					
BSR		5	1	Op Code Address+1	1	Offset
			2	FFFF	1	Restart Address (LSB)
			3	Stack Pointer	0	Return Address (LSB)
			4	Stack Pointer−1	0	Return Address (MSB)
			5	Branch Address	1	First Op Code of Subroutine

■ LOW POWER CONSUMPTION MODE

The HD6303R has two low power consumption modes; sleep and standby mode.

• Sleep Mode

On execution of SLP instruction, the MPU is brought to the sleep mode. In the sleep mode, the CPU stops its operation, but the contents of the registers in the CPU are retained. In this mode, the peripherals of CPU will remain active. So the operations such as transmit and receive of the SCI data and counter may keep in operation. In this mode, the power consumption is reduced to about 1/6 the value of a normal operation.

The escape from this mode can be done by interrupt, $\overline{RES}$, $\overline{STBY}$. The $\overline{RES}$ resets the MPU and the $\overline{STBY}$ brings it into the standby mode (This will be mentioned later). When interrupt is requested to the CPU and accepted, the sleep mode is released, then the CPU is brought in the operation mode and jumps to the interrupt routine. When the CPU has masked the interrupt, after recovering from the sleep mode, the next instruction of SLP starts to execute. However, in such a case that the timer interrupt is inhibited on the timer side, the sleep mode cannot be released due to the absence of the interrupt request to the CPU.

This sleep mode is available to reduce an average power consumption in the applications of the HD6303R which may not be always running.

• Standby Mode

Bringing $\overline{STBY}$ "Low", the CPU becomes reset and all clocks of the HD6303R become inactive. It goes into the standby mode. This mode remarkably reduces the power consumptions of the HD6303R.

In the standby mode, if the HD6303R is continuously supplied with power, the contents of RAM is retained. The standby mode should escape by the reset start. The following is the typical application of this mode.

First, $\overline{NMI}$ routine stacks the CPU's internal information and the contents of SP in RAM, disables RAME bit of RAM control register, sets the standby bit, and then goes into the standby mode. If the standby bit keeps set on reset start, it means that the power has been kept during stand-by mode and the contents of RAM is normally guaranteed. The system recovery may be possible by returning SP and bringing into the condition before the standby mode has started. The timing relation for each line in this application is shown in Figure 21.

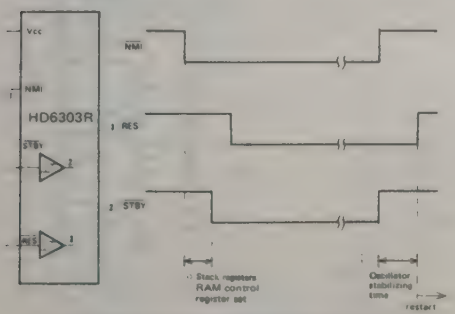


Figure 21 Standby Mode Timing

■ ERROR PROCESSING

When the HD6303R fetches an undefined instruction or fetches an instruction from unusable memory area, it generates the highest priority internal interrupt, that may protect from system upset due to noise or a program error.

● Op-Code Error

Fetching an undefined op-code, the HD6303R will stack the CPU register as in the case of a normal interrupt and vector to the TRAP ($FFEE, $FFEF), that has a second highest priority ($\overline{RES}$ is the highest).

● Address Error

When an instruction is fetched from other than a resident RAM, or an external memory area, the CPU starts the same interrupt as op-code error. In the case which the instruction is fetched from external memory area and that area is not usable, the address error can not be detected.

The address which cause address error are shown in Table 13.

This feature is applicable only to the instruction fetch, not to normal read/write of data accessing.

Transitions among the active mode, sleep mode, standby mode and reset are shown in Figure 22.

Figures 23, 24 show a system configuration.

The system flow chart of HD6303R is shown in Figure 25.

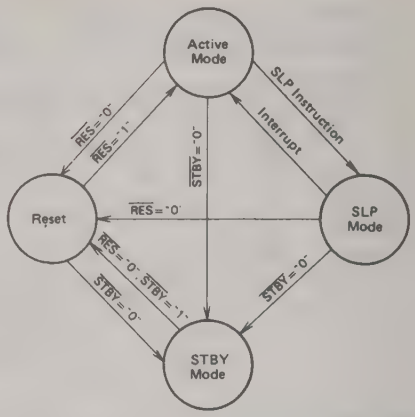


Figure 22 Transitions among Active Mode, Standby Mode, Sleep Mode, and Reset

Table 13 Address Error

Address Error
$0000 ~ $001F

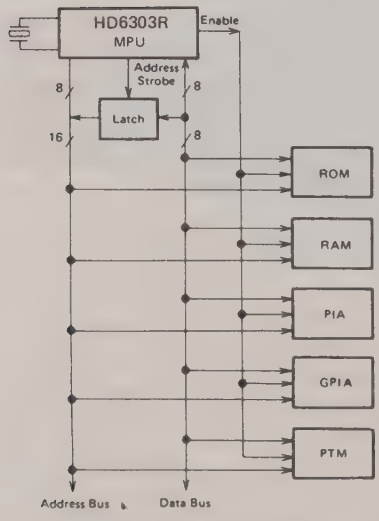


Figure 23 HD6303R MPU Multiplexed Mode

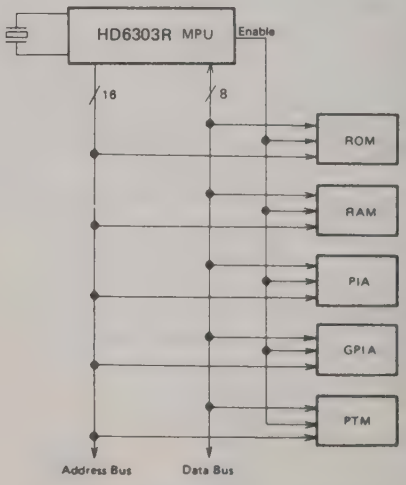


Figure 24 HD6303R MPU Non-Multiplexed Mode

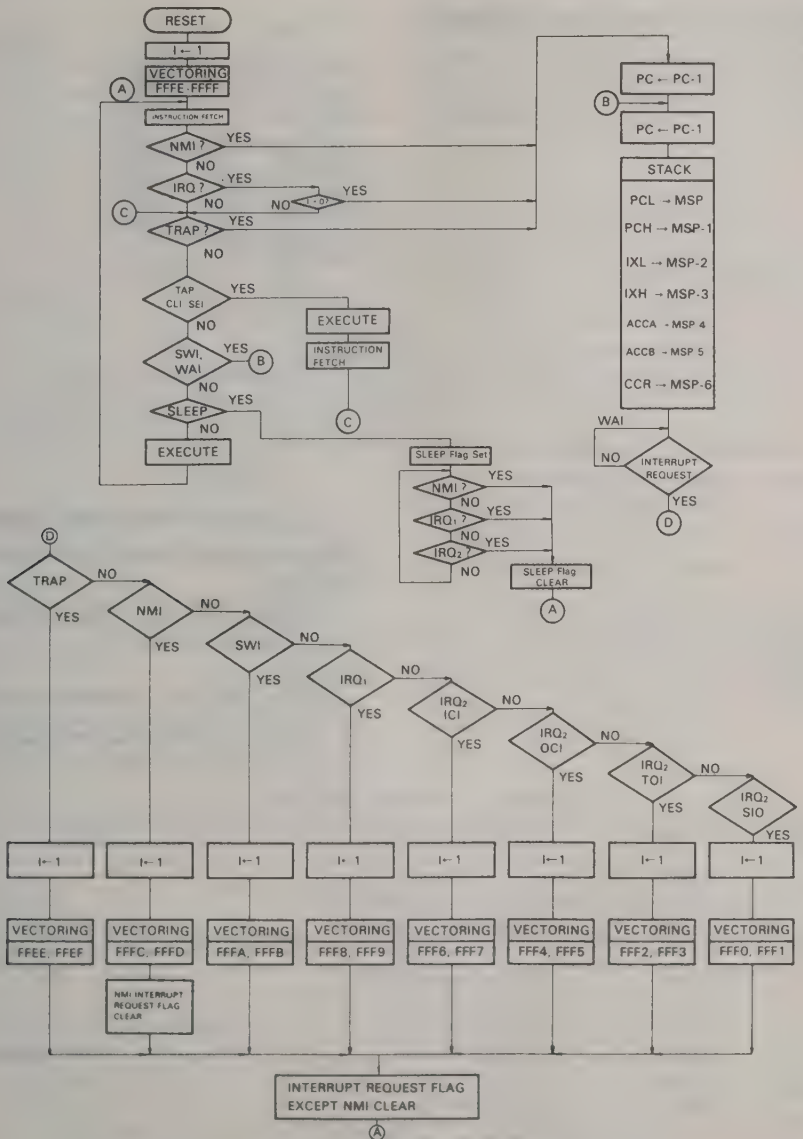


Figure 25 HD6303R System Flow Chart

■ PRECAUTION TO THE BOARD DESIGN OF OSCILLATION CIRCUIT

As shown in Fig. 26, there is a case that the cross talk disturbs the normal oscillation if signal lines are put near the oscillation circuit. When designing a board, pay attention to this. Crystal and C_L must be put as near the HD6303R as possible.

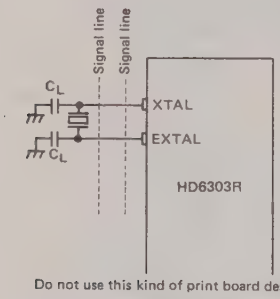


Figure 26 Precaution to the boad design of oscillation circuit

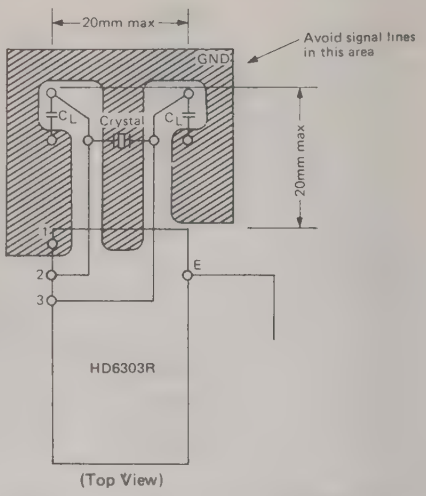


Fig. 27 Example of Oscillation Circuits in Board Design

■ PIN CONDITIONS AT SLEEP AND STANDBY STATE

• Sleep State

The conditions of power supply pins, clock pins, input pins and E clock pin are the same as those of operation. Refer to Table 14 for the other pin conditions.

• Standby State

Only power supply pins and $\overline{STBY}$ are active. As for the clock pin EXTAL, its input is fixed internally so the MPU is not influenced by the pin conditions. XTAL is in "1" output. All the other pins are in high impedance.

Table 14 Pin Condition in Sleep State

Pin	Mode	Non Multiplexed Mode	Multiplexed Mode
$P_{20} \sim P_{24}$	Function	I/O Port	I/O Port
	Condition	Keep the condition just before sleep	←
$A_0/P_{10} \sim A_7/P_{17}$	Function	Address Bus ($A_0 \sim A_7$)	I/O Port
	Condition	Output "1"	Keep the condition just before sleep
$A_8 \sim A_{15}$	Function	Address Bus ($A_8 \sim A_{15}$)	Address Bus ($A_8 \sim A_{15}$)
	Condition	Output "1"	←
$D_0/A_0 \sim D_7/A_7$	Function	Data Bus ($D_0 \sim D_7$)	$\overline{E}$: Address Bus ($A_0 \sim A_7$), E: Data Bus
	Condition	High Impedance	$\overline{E}$: Output "1", E: High Impedance
R/$\overline{W}$	Function	R/$\overline{W}$ Signal	R/$\overline{W}$ Signal
	Condition	Output "1"	←
AS		–	Output AS

Table 15 Pin Condition during RESET

Pin \ Mode	Non-Multiplexed Mode	Multiplexed Mode
$P_{20} \sim P_{24}$	High Impedance	←
$A_0/P_{10} \sim A_7/P_{17}$	High Impedance	←
$A_8 \sim A_{15}$	High Impedance	←
$D_0/A_0 \sim D_7/A_7$	High Impedance	$\overline{E}$: "1" Output E : High Impedance
R/$\overline{W}$	"1" Output	←
AS	$\overline{E}$: "1" Output E : "0" Output	←

(Note) In the multiplexed mode, the data bus is set to "1" output state during E = "1" and it causes the conflict with the output of external memory. Following 1 and 2 should be done to avoid the conflict;
(1) Construct the system that disables the external memory during reset.
(2) Add 4.7 kΩ pull-down resistance to the AS pin to make AS pin "0" level during E = "1". This operation makes the data bus high impedance state.

■ DIFFERENCE BETWEEN HD6303 AND HD6303R

The HD6303R is an upgraded version of the HD6303. The difference between HD6303 and HD6303R is shown in Table 16.

Table 16 Difference between HD6303 and HD6303R

Item	HD6303	HD6303R
Operating Mode	Mode 2: Not defined	Mode 2: Multiplexed Mode (Equivalent to Mode 4)
Electrical Characteristics	The electrical characteristics of 2MHz version (B version) are not specified.	Some characteristics are improved. The 2MHz version is guaranteed.
Timer	Has problem in output compare function. (Can be avoided by software.)	The problem is solved.

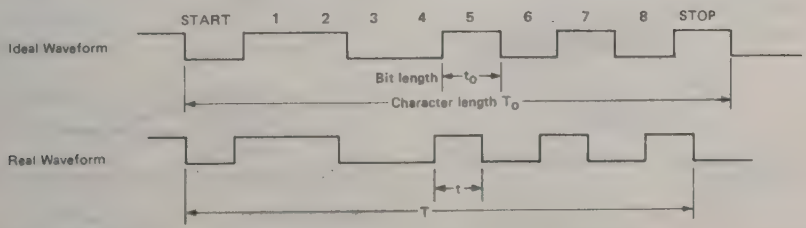
■ RECEIVE MARGIN OF THE SCI

Receive margin of the SCI contained in the HD6303R is shown in Table 17.

Note: SCI = Serial Communication Interface

Table 17

	Bit distortion tolerance (t−to) /to	Character distortion tolerance (T−To) /To
HD6303R	±37.5%	+3.75% −2.5%



■ APPLICATION NOTE FOR HIGH SPEED SYSTEM DESIGN USING THE HD6303R

This note describes the solutions of the potential problem caused by noise generation in the system using the HD6303R.

The CMOS ICs and LSIs featured by low power consumption and high noise immunity are generally considered to be enough with simply designed power source and the GND line.

But this does not apply to the applications configured of high speed system or of high speed parts. Such high speed system may have a chance to work incorrectly because of the noise

by the transient current generated during switching. One of example is a system in which the HD6303R directly accesses high speed memory such as the HM6264. The noise generation owing to the over current (Sometimes it may be several hundreds mA for peak level.) during switching may cause data write error.

This noise problem may be observed only at the Expanded Mode (Mode 1, 2, 4, 5 and 6) of the HD6303R.

Assuming the HD6303R is used as CPU in a system.

I. Noise Occurrence

If the HD6303R is connected to high speed RAM, a write error may occur. As shown in Fig. 28, the noise is generated in address bus during write cycle and data is written into an unexpected address from the HD6303R. This phenomenon causes random failures in systems whose data bus load capacitance exceeds the specification value (90 pF max.) and/or the impedance of the GND line is high.

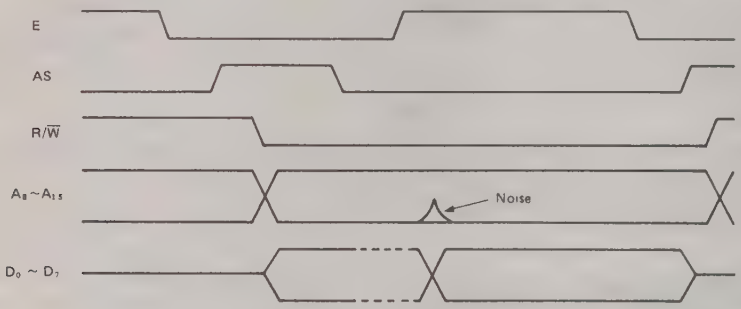


Fig. 28 Noise Occurrence in address bus during write cycle

If the data bus $D_0 \sim D_7$ changes from "FF" to "00", extremely large transient current flows through the GND line. Then the noise is generated on the LSI's V_{SS} pins proportioning to the transient current and to the impedance [Zg] of the GND line.

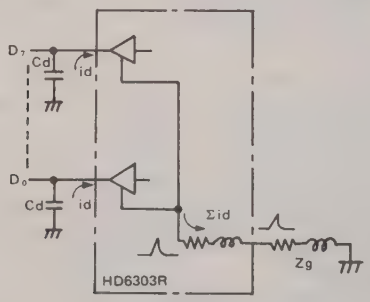
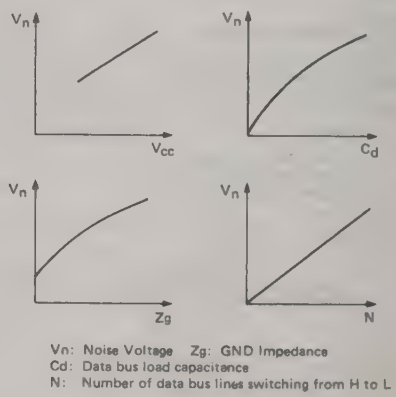


Fig. 29 Noise Source

This noise level, V_n, appears on all output pins on the LSI including the address bus.

Fig. 30 shows the dependency of the noise voltage on the each parameter.



Vn: Noise Voltage Zg: GND Impedance
Cd: Data bus load capacitance
N: Number of data bus lines switching from H to L

Fig. 30 Dependency of the noise voltage on each parameter

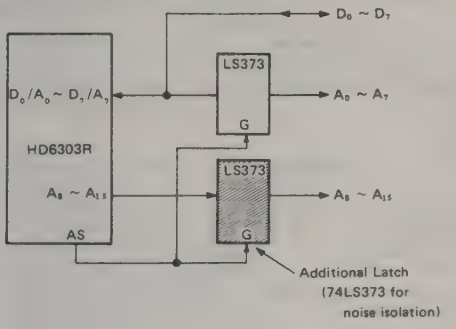
II. Noise Protection

To avoid the noise on the address bus during the system operation mentioned before, there are two solutions as follows:

The one method is to isolate the HD6303R from peripheral devices so that peripherals are not affected by the noise. The other is to reduce noise level to the extent of not affecting peripherals using analog method.

1. Noise Isolation

Addresses should be latched at the negative edge of the AS signal or at the positive edge of the E signal. The 74LS373 is often used in this case.



2. Noise Reduction

As the noise level depends on each parameter such Cd, V_{CC}, Zg, the noise level can be reduced to the allowable level by controlling those analog parameters.

(a) Transient Current Reduction

(1) Reduce the data bus load capacitance. If large load capacitance is expected, a bus buffer should be inserted.

(2) Lower the power supply voltage V_{CC} within specification.

(3) Increase a time constant at transient state by inserting a resistor (100 ~ 200Ω) to Data Buses in series to keep noise level down.
Table 18 shows the relationship between a series resistor and noise level or a resistor and DC/AC characteristics.

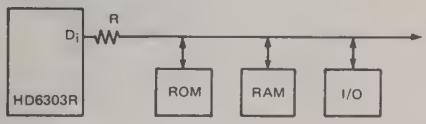


Table 18.

Item		Resistor	No	100Ω	200Ω
Noise Voltage Level			See Fig. 31		
DC Characteristics		I_{OL}	1.6 mA	1.6 mA	1.0 mA
AC Characteristics	f = 1 MHz		No change		
	f = 1.5 MHz	t_{ADL}	190 ns	190 ns	210 ns
		t_{ACCM}	395 ns	395 ns	375 ns
	f = 2 MHz	t_{ADL}	160 ns	180 ns	200 ns
		t_{ASL}	20 ns	20 ns	0 ns
		t_{ACCM}	270 ns	250 ns	230 ns

Fig. 31 shows an example of the dependency of the noise voltage on the load capacitance of the data bus.*

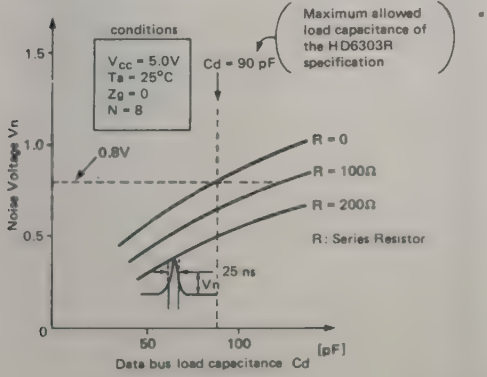


Fig. 31

*Note: (The value of series resistor should be carefully selected because it heavily depends on each parameter of actual application system.)

Fig. 32 shows the typical wave form of the noise.

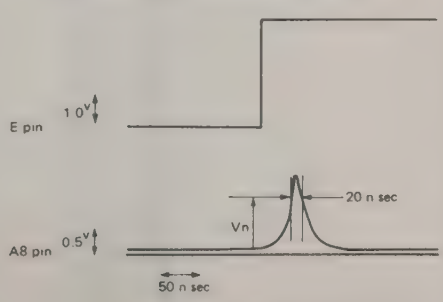


Fig. 32

(b) Reduction of GND line impedance
(1) Widen the GND line width on the PC board.
(2) Place the HD6303R close by power source.
(3) Insert a bypass capacitor between the V_{CC} line and the GND of the HD6303R. A tantalum capacitor (about 0.1μF) is effective on the reduction.

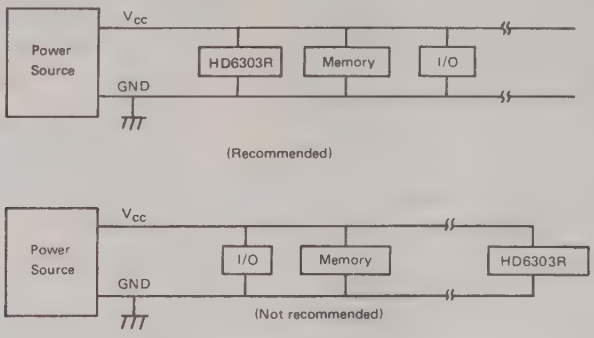


Fig. 33 Layout of the HD6303R on the PC board

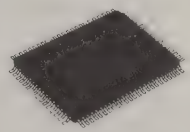
HD6303X, HD63A03X, HD63B03X
CMOS MPU (Micro Processing Unit)

The HD6303X is a CMOS 8-bit micro processing unit (MPU) which includes a CPU compatible with the HD6301V1, 192 bytes of RAM, 24 parallel I/O pins, a Serial Communication Interface (SCI) and two timers on chip.

HD6303XP, HD63A03XP, HD63B03XP

(DP-64S)

HD6303XF, HD63A03XF, HD63B03XF



(FP-80)

■ FEATURES

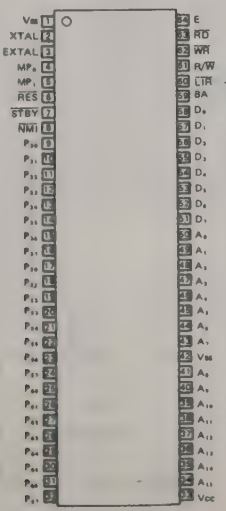
- Instruction Set Compatible with the HD6301V1
- 192 Bytes of RAM
- 24 Parallel I/O Pins
  16 I/O Pins-Port 2, 6
  8 Input Pins-Port 5
- Darlington Transistor Drive (Port 2, 6)
- 16-Bit Programmable Timer
  Input Capture Register x 1
  Free Running Counter x 1
  Output Compare Register x 2
- 8-Bit Reloadable Timer
  External Event Counter Square Wave Generation
- Serial Communication Interface
- Memory Ready
- Halt
- Error-Detection (Address Trap, Op-Code Trap)
- Interrupts . . . 3 External, 7 Internal
- Up to 65k Bytes Address Space
- Low Power Dissipation Mode
  Sleep Mode
  Standby Mode
- Minimum Instruction Execution Time -0.5μs (f = 2.0 MHz)
- Wide Range of Operation

V_{CC} = 3 ~ 6V (f = 0.1 ~ 0.5 MHz).

V_{CC} = 5V±10% { f = 0.1 ~ 1.0 MHz; HD6303X
f = 0.1 ~ 1.5 MHz; HD63A03X
f = 0.1 ~ 2.0 MHz; HD63B03X

■ PIN ARRANGEMENT

- HD6303XP, HD63A03XP, HD63B03XP

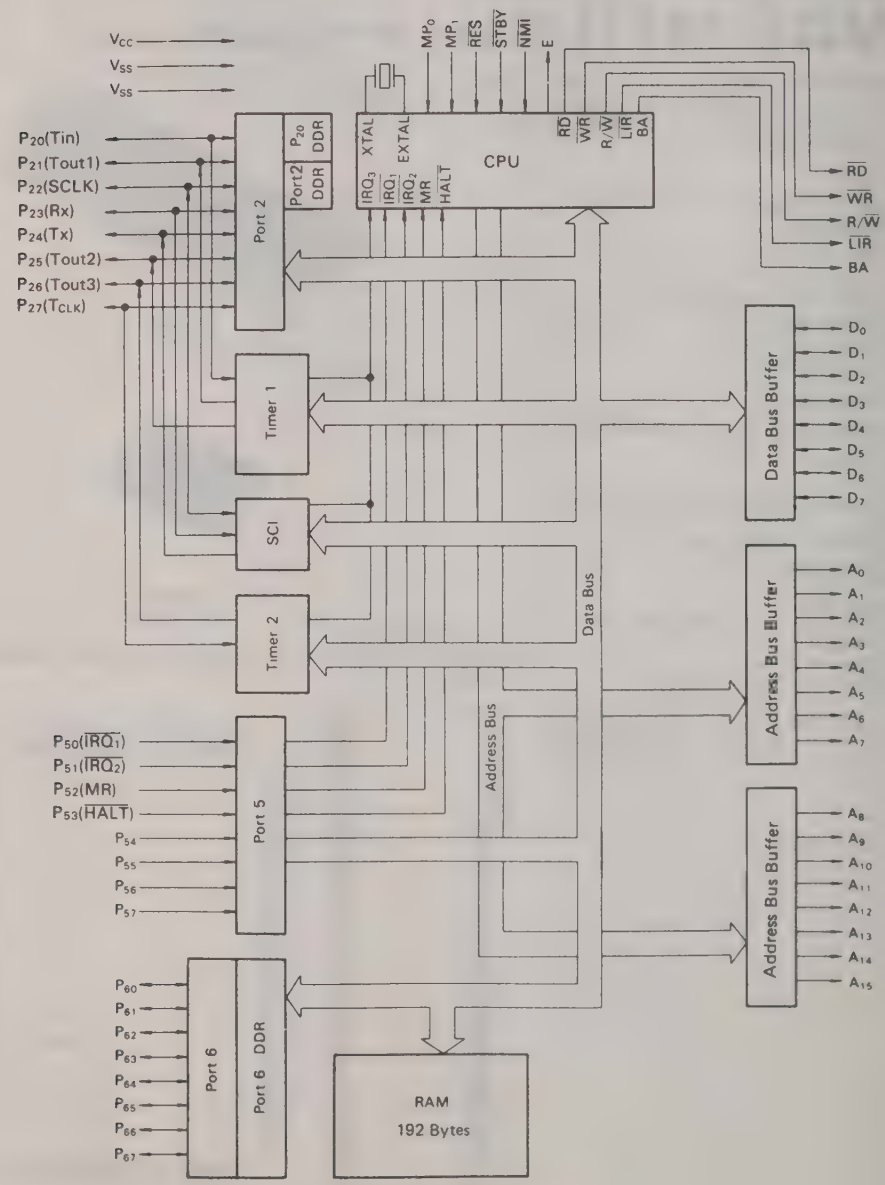


(Top View)

- HD6303XF, HD63A03XF, HD63B03XF

(Top View)

■ **BLOCK DIAGRAM**



■ ABSOLUTE MAXIMUM RATINGS

Item	Symbol	Value	Unit
Supply Voltage	V_{CC}	-0.3 ~ +7.0	V
Input Voltage	V_{in}	-0.3 ~ V_{CC}+0.3	V
Operating Temperature	T_{opr}	0 ~ +70	°C
Storage Temperature	T_{stg}	-55 ~ +150	°C

(NOTE) This product has protection circuits in input terminal from high static electricity voltage and high electric field. But be careful not to apply overvoltage more than maximum ratings to these high input impedance protection circuits. To assure the normal operation, we recommend V_{in}, V_{out}: $V_{SS} \leqq (V_{in}$ or $V_{out}) \leqq V_{CC}$.

■ ELECTRICAL CHARACTERISTICS

• DC CHARACTERISTICS (V_{CC} = 5.0V±10%, V_{SS} = 0V, Ta = 0 ~ +70°C, unless otherwise noted.)

Item		Symbol	Test Condition	min	typ	max	Unit
Input "High" Voltage	$\overline{RES}$, $\overline{STBY}$	V_{IH}		V_{CC}-0.5	–	V_{CC} +0.3	V
	EXTAL			V_{CC}x0.7	–		
	Other Inputs			2.0	–		
Input "Low" Voltage	All Inputs	V_{IL}		-0.3	–	0.8	V
Input Leakage Current	$\overline{NMI}$, $\overline{RES}$, $\overline{STBY}$, MP_0, MP_1, Port 5	$\|I_{in}\|$	V_{in} = 0.5 ~ V_{CC}-0.5V	–	–	1.0	μA
Three State (off-state) Leakage Current	A_0~A_{15}, D_0~D_7, $\overline{RD}$, $\overline{WR}$, R/$\overline{W}$, Port 2, Port 6	$\|I_{TSI}\|$	V_{in} = 0.5 ~ V_{CC}-0.5V	–	–	1.0	μA
Output "High" Voltage	All Outputs	V_{OH}	I_{OH} = -200μA	2.4	–	–	V
			I_{OH} = -10μA	V_{CC}-0.7	–	–	V
Output "Low" Voltage	All Outputs	V_{OL}	I_{OL} = 1.6mA	–	–	0.4	V
Darlington Drive Current	Ports 2, 6	$-I_{OH}$	Vout = 1.5V	1.0	–	10.0	mA
Input Capacitance	All Inputs	C_{in}	V_{in} = 0V, f = 1MHz, Ta = 25°C	–	–	12.5	pF
Standby Current	Non Operation	I_{STB}		–	3.0	15.0	μA
Current Dissipation*		I_{SLP}	Sleeping (f = 1MHz**)	–	1.5	3.0	mA
			Sleeping (f = 1.5MHz**)	–	2.3	4.5	mA
			Sleeping (f = 2MHz**)	–	3.0	6.0	mA
		I_{CC}	Operating (f = 1MHz**)	–	7.0	10.0	mA
			Operating (f = 1.5MHz**)	–	10.5	15.0	mA
			Operating (f = 2MHz**)	–	14.0	20.0	mA
RAM Standby Voltage		V_{RAM}		2.0	–	–	V

* V_{IH} min = V_{CC}-1.0V, V_{IL} max = 0.8V ; All output terminals are at no load.

** Current Dissipation of the operating or sleeping condition is proportional to the operating frequency. So the typ. or max. values about Current Dissipations at x MHz operation are decided according to the following formula;

typ. value (f = x MHz) = typ. value (f = 1MHz) x x
max. value (f = x MHz) = max. value (f = 1MHz) x x
(both the sleeping and operating)

• AC CHARACTERISTICS (V_{CC} = 5.0V±10%, V_{SS} = 0V, T_a = 0 ~ +70°C, unless otherwise noted.)

BUS TIMING

Item		Symbol	Test Condition	HD6303X			HD63A03X			HD63B03X			Unit
				min	typ	max	min	typ	max	min	typ	max	
Cycle Time		t_{cyc}	Fig. 1	1	–	10	0.666	–	10	0.5	–	10	μs
Enable Rise Time		t_{Er}		–	–	25	–	–	25	–	–	25	ns
Enable Fall Time		t_{Ef}		–	–	25	–	–	25	–	–	25	ns
Enable Pulse Width "High" Level*		PW_{EH}		450	–	–	300	–	–	220	–	–	ns
Enable Pulse Width "Low" Level*		PW_{EL}		450	–	–	300	–	–	220	–	–	ns
Address, R/$\overline{W}$ Delay Time*		t_{AD}		–	–	250	–	–	190	–	–	160	ns
Data Delay Time	Write	t_{DDW}		–	–	200	–	–	160	–	–	120	ns
Data Set-up Time	Read	t_{DSR}		80	–	–	70	–	–	70	–	–	ns
Address, R/$\overline{W}$ Hold Time*		t_{AH}		80	–	–	50	–	–	35	–	–	ns
Data Hold Time	Write*	t_{HW}		80	–	–	50	–	–	40	–	–	ns
	Read	t_{HR}		0	–	–	0	–	–	0	–	–	ns
$\overline{RD}$, $\overline{WR}$ Pulse Width*		PW_{RW}		450	–	–	300	–	–	220	–	–	ns
$\overline{RD}$, $\overline{WR}$ Delay Time		t_{RWD}		–	–	40	–	–	40	–	–	40	ns
$\overline{RD}$, $\overline{WR}$ Hold Time		t_{HRW}		–	–	30	–	–	30	–	–	25	ns
$\overline{LIR}$ Delay Time		t_{DLR}		–	–	200	–	–	160	–	–	120	ns
$\overline{LIR}$ Hold Time		t_{HLR}		10	–	–	10	–	–	10	–	–	ns
MR Set-up Time*		t_{SMR}	Fig. 2	400	–	–	280	–	–	230	–	–	ns
MR Hold Time*		t_{HMR}		–	–	90	–	–	40	–	–	0	ns
E Clock Pulse Width at MR		PW_{EMR}		–	–	9	–	–	9	–	–	9	μs
Processor Control Set-up Time		t_{PCS}	Fig. 3, 10, 11	200	–	–	200	–	–	200	–	–	ns
Processor Control Rise Time		t_{PCr}	Fig. 2, 3	–	–	100	–	–	100	–	–	100	ns
Processor Control Fall Time		t_{PCf}		–	–	100	–	–	100	–	–	100	ns
BA Delay Time		t_{BA}	Fig. 3	–	–	250	–	–	190	–	–	160	ns
Oscillator Stabilization Time		t_{RC}	Fig. 11	20	–	–	20	–	–	20	–	–	ms
Reset Pulse Width		PW_{RST}		3	–	–	3	–	–	3	–	–	t_{cyc}

* These timings change in approximate proportion to t_{cyc}. The figures in this characteristics represent those when t_{cyc} is minimum (= in the highest speed operation).

PERIPHERAL PORT TIMING

Item		Symbol	Test Condition	HD6303X			HD63A03X			HD63B03X			Unit
				min	typ	max	min	typ	max	min	typ	max	
Peripheral Data Set-up Time	Ports 2, 5, 6	t_{PDSU}	Fig. 5	200	–	–	200	–	–	200	–	–	ns
Peripheral Data Hold Time	Ports 2, 5, 6	t_{PDH}	Fig. 5	200	–	–	200	–	–	200	–	–	ns
Delay Time (Enable Negative Transition to Peripheral Data Valid)	Ports 2, 6	t_{PWD}	Fig. 6	–	–	300	–	–	300	–	–	300	ns

TIMER, SCI TIMING

Item		Symbol	Test Condition	HD6303X			HD63A03X			HD63B03X			Unit
				min	typ	max	min	typ	max	min	typ	max	
Timer 1 Input Pulse Width		t_{PWT}	Fig. 8	2.0	–	–	2.0	–	–	2.0	–	–	t_{cyc}
Delay Time (Enable Positive Transition to Timer Output)		t_{TOD}	Fig. 7	–	–	400	–	–	400	–	–	400	ns
SCI Input Clock Cycle	Async. Mode	t_{Scyc}	Fig. 8	1.0	–	–	1.0	–	–	1.0	–	–	t_{cyc}
	Clock Sync.		Fig. 4, 8	2.0	–	–	2.0	–	–	2.0	–	–	t_{cyc}
SCI Transmit Data Delay Time (Clock Sync. Mode)		t_{TXD}	Fig. 4	–	–	200	–	–	200	–	–	200	ns
SCI Receive Data Set-up Time (Clock Sync. Mode)		t_{SRX}		290	–	–	290	–	–	290	–	–	ns
SCI Receive Data Hold Time (Clock Sync. Mode)		t_{HRX}		100	–	–	100	–	–	100	–	–	ns
SCI Input Clock Pulse Width		t_{PWSCK}	Fig. 8	0.4	–	0.6	0.4	–	0.6	0.4	–	0.6	t_{Scyc}
Timer 2 Input Clock Cycle		t_{tcyc}		2.0	–	–	2.0	–	–	2.0	–	–	t_{cyc}
Timer 2 Input Clock Pulse Width		t_{PWTCK}		200	–	–	200	–	–	200	–	–	ns
Timer 1•2, SCI Input Clock Rise Time		t_{CKr}		–	–	100	–	–	100	–	–	100	ns
Timer 1•2, SCI Input Clock Fall Time		t_{CKf}		–	–	100	–	–	100	–	–	100	ns

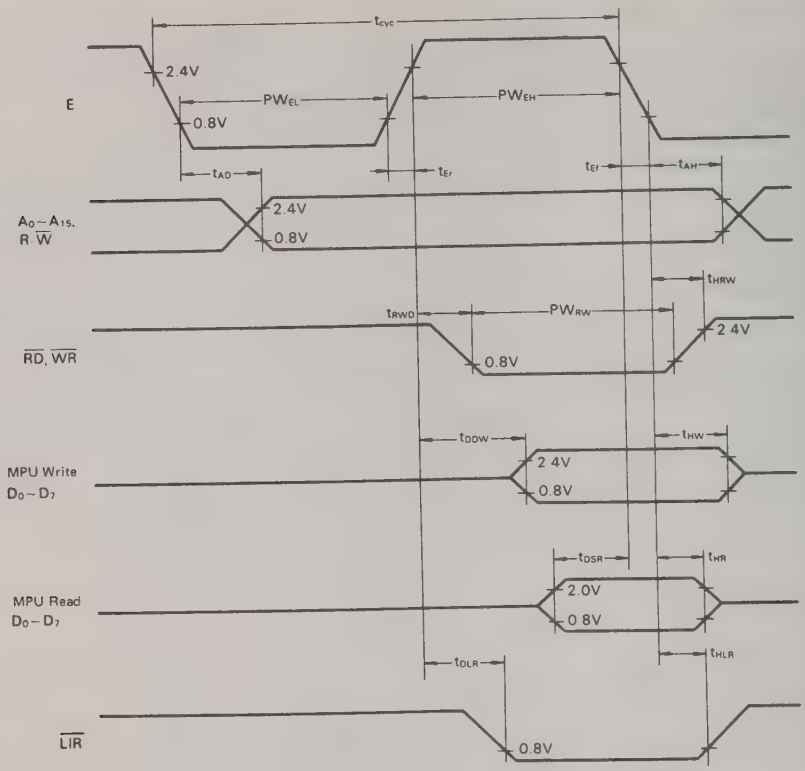


Figure 1 Bus Timing

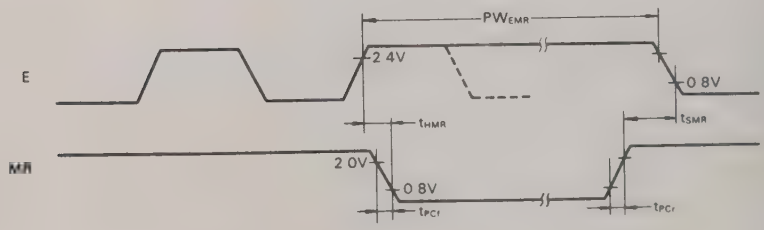


Figure 2 Memory Ready and E Clock Timing

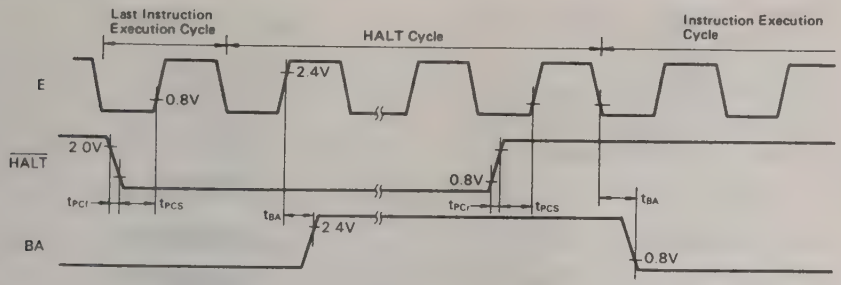


Figure 3 $\overline{\text{HALT}}$ and BA Timing

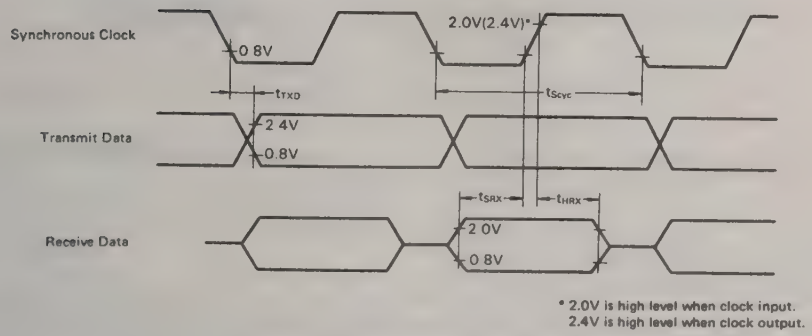


Figure 4 SCI Clocked Synchronous Timing

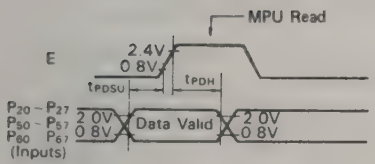


Figure 5 Port Data Set-up and Hold Times (MPU Read)

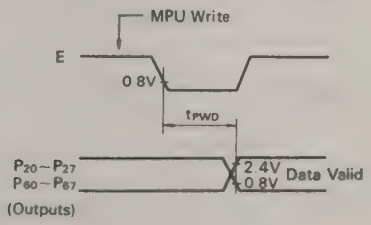
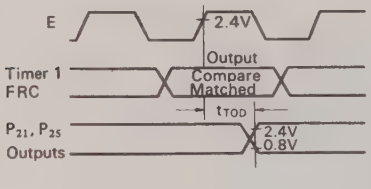
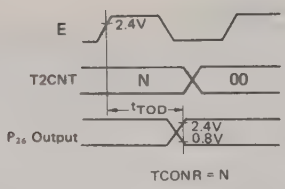


Figure 6 Port Data Delay Times (MPU Write)

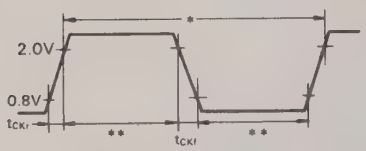


(a) Timer 1 Output Timing



(b) Timer 2 Output Timing

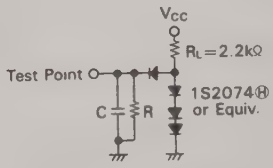
Figure 7 Timer Output Timing



* Timer 2; t_{cyc} SCI ; t_{Scyc}

** Timer 1; t_{PWT}
Timer 2; t_{PWTCK}
SCI ; t_{PWSCK}

Figure 8 Timer 1·2, SCI Input Clock Timing



C=90pF for D_0~D_7, A_0~A_{15}, E
=30pF for Port 2, Port 6, $\overline{RD}$, $\overline{WR}$, R/$\overline{W}$, BA, $\overline{LIR}$
R=12kΩ

Figure 9 Bus Timing Test Loads (TTL Load)

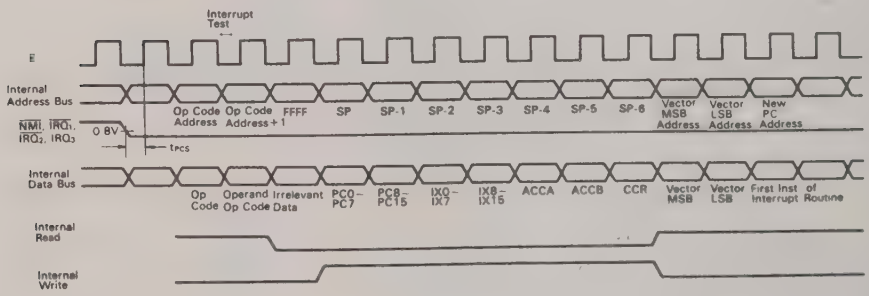


Figure 10 Interrupt Sequence

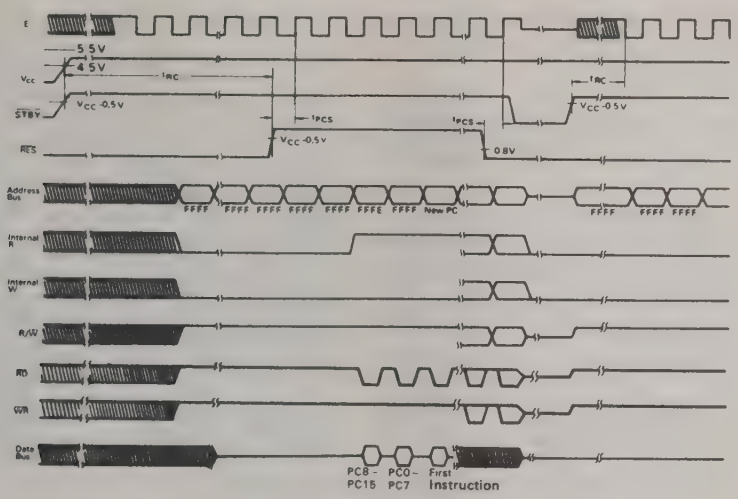


Figure 11 Reset Timing

■ FUNCTIONAL PIN DESCRIPTION

• V_{CC}, V_{SS}

V_{CC} and V_{SS} provide power to the MPU with 5V±10% supply. In the case of low speed operation (fmax = 500kHz), the MPU can operate with three through six volts. Two V_{SS} pins should be tied to ground.

• XTAL, EXTAL

These two pins interface with an AT-cut parallel resonant crystal. Divide-by-four circuit is on chip, so if 4MHz crystal oscillator is used, the system clock is 1MHz for example.

AT Cut Parallel Resonant Crystal Oscillator

C_O=7pF max
R_S=60Ω max

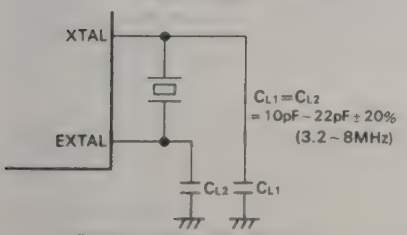


Figure 12 Crystal Interface

EXTAL pin can be drived by the external clock of 45 to 55% duty, and one fourth frequency of the external clock is produced in the LSI. The external clock frequency should be less than four times of the maximum operable frequency. When using the external clock, XTAL pin should be open. Fig. 12 shows an example of the crystal interface. The crystal and C_{L1}, C_{L2} should be mounted as close as possible to XTAL and EXTAL pins. Any line must not cross the line between the crystal oscillator and XTAL, EXTAL.

• $\overline{STBY}$

This pin makes the MPU standby mode. In "Low" level, the oscillation stops and the internal clock is stabilized to make reset condition. To retain the contents of RAM at standby mode, "0" should be written into RAM enable bit (RAME). RAME is the bit 6 of the RAM/port 5 control register at $0014. RAM is disabled by this operation and its contents is sustained.

Refer to "LOW POWER DISSIPATION MODE" for the standby mode.

• Reset ($\overline{RES}$)

This pin resets the MPU from power OFF state and provides a startup procedure. During power-on, $\overline{RES}$ pin must be held "Low" level for at least 20ms.

The CPU registers (accumulator, index register, stack pointer, condition code register except for interrupt mask bit), RAM and the data register of a port are not initialized during reset, so their contents are unknown in this procedure.

To reset the MPU during operation, $\overline{RES}$ should be held "Low" for at least 3 system-clock cycles. At the 3rd cycle during "Low" level, all the address buses become "High". When $\overline{RES}$ remains "Low", the address buses keep "High". If $\overline{RES}$ becomes "High", the MPU starts the next operation.

(1) Latch the value of the mode program pins; MP_0 and MP_1.
(2) Initialize each internal register (Refer to Table 3).
(3) Set the interrupt mask bit. For the CPU to recognize the maskable interrupts $\overline{IRQ_1}$, $\overline{IRQ_2}$ and IRQ_3, this bit should be cleared in advance.
(4) Put the contents (= start address) of the last two addresses ($FFFE, $FFFF) into the program counter and start the program from this address. (Refer to Table 1).

*The MPU is usable to accept a reset input until the clock

becomes normal oscillation after power on (max. 20ms). During this transient time, the MPU and I/O pins are undefined. Please be aware of this for system designing.

- **Enable (E)**

This pin provides a TTL-compatible system clock to external circuits. Its frequency is one fourth that of the crystal oscillator or external clock. This pin can drive one TTL load and 90pF capacitance.

- **Non-Maskable Interrupt ($\overline{NMI}$)**

When the falling edge of the input signal is detected at this pin, the CPU begins non-maskable interrupt sequence internally. As well as the IRQ mentioned below, the instruction being executed at $\overline{NMI}$ signal detection will proceed to its completion. The interrupt mask bit of the condition code register doesn't affect non-maskable interrupt at all.

When starting the acknowledge to the $\overline{NMI}$, the contents of the program counter, index register, accumulators and condition code register will be saved onto the stack. Upon completion of this sequence, a vector is fetched from $FFFC and $FFFD to transfer their contents into the program counter and branch to the non-maskable interrupt service routine.

(Note) After reset start, the stack pointer should be initialized on an appropreate memory area and then the falling edge should be input to $\overline{NMI}$ pin.

- **Interrupt Request ($\overline{IRQ_1}$, $\overline{IRQ_2}$)**

These are level-sensitive pins which request an internal interrupt sequence to the CPU. At interrupt request, the CPU will complete the current instruction before its request acknowledgement. Unless the interrupt mask in the condition code register is set, the CPU starts an interrupt sequence; if set, the interrupt request will be ignored. When the sequence starts, the contents of the program counter, index register, accumulators and condition code register will be saved onto the stack, then the CPU sets the interrupt mask bit and will not acknowledge the maskable request. During the last cycle, the CPU fetches vectors depicted in Table 1 and transfers their contents to the program counter and branches to the service routine.

The CPU uses the external interrupt pins, $\overline{IRQ_1}$ and $\overline{IRQ_2}$, also as port pins P_{50} and P_{51}, so it provides an enable bit to Bit 0 and 1 of the RAM port 5 control register at $0014. Refer to "RAM/PORT 5 CONTROL REGISTER" for the details.

When one of the internal interrupts, ICI, OCI, TOI, CMI or SIO is generated, the CPU produces internal interrupt signal (IRQ_3). IRQ_3 functions just the same as $\overline{IRQ_1}$ or $\overline{IRQ_2}$ except for its vector address. Fig. 13 shows the block diagram of the interrupt circuit.

Table 1 Interrupt Vector Memory Map

Priority	Vector		Interrupt
	MSB	LSB	
Highest	FFFE	FFFF	$\overline{RES}$
	FFEE	FFEF	TRAP
	FFFC	FFFD	$\overline{NMI}$
	FFFA	FFFB	SWI (Software Interrupt)
	FFF8	FFF9	$\overline{IRQ_1}$
	FFF6	FFF7	ICI (Timer 1 Input Capture)
	FFF4	FFF5	OCI (Timer 1 Output Compare 1, 2)
	FFF2	FFF3	TOI (Timer 1 Overflow)
	FFEC	FFED	CMI (Timer 2 Counter Match)
	FFEA	FFEB	$\overline{IRQ_2}$
Lowest	FFF0	FFF1	SIO (RDRF+ORFE+TDRE)

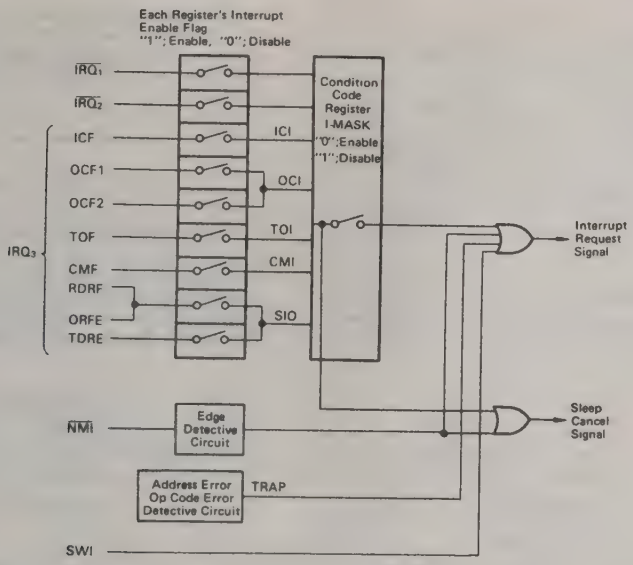


Figure 13 Interrupt Circuit Block Diagram

- **Mode Program (MP_0, MP_1)**

To operate MPU, MP_0 pin should be connected to "High" level and MP_1 should be connected to "Low" level (refer to Fig. 15).

- **Read/Write (R/$\overline{W}$)**

This signal, usually be in read state ("High"), shows whether the CPU is in read ("High") or write ("Low") state to the peripheral or memory devices. This can drive one TTL load and 30pF capacitance.

- **$\overline{RD}$, $\overline{WR}$**

These signals show active low outputs when the CPU is reading/writing to the peripherals or memories. This enables the CPU easy to access the peripheral LSI with $\overline{RD}$ and $\overline{WR}$ input pins. These pins can drive one TTL load and 30pF capacitance.

- **Load Instruction Register ($\overline{LIR}$)**

This signal shows the instruction opecode being on data bus (active low). This pin can drive one TTL load and 30pF capacitance.

- **Memory Ready (MR; P_{52})**

This is the input control signal which stretches the system clock's "High" period to access low-speed memories. During this signal is in "High", the system clock operates in normal sequence. But this signal in "Low", the "High" period of the system clock will be stretched depending on its "Low" level duration in integral multiples of the cycle time. This allows the CPU to interface with low-speed memories (see Fig. 2). Up to 9 μs can be stretched.

During internal address space access or nonvalid memory access, MR is prohibited internally to prevent decrease of operation speed. Even in the halt state, MR can also stretch "High" period of system clock to allow peripheral devices to access low-speed memories. As this signal is used also as P_{52}, an enable bit is provided at bit 2 of the RAM/port 5 control register at $0014. Refer to "RAM/PORT 5 CONTROL REGISTER" for more details.

- **Halt ($\overline{HALT}$; P_{53})**

This is an input control signal to stop instruction execution and to release buses. When this signal switches to "Low", the CPU stops to enter into the halt state after having executed the present instruction. When entering into the halt state, it makes BA (P_{74}) "High" and also an address bus, data bus, $\overline{RD}$, $\overline{WR}$, R/$\overline{W}$ high impedance. When an interrupt is generated in the halt state, the CPU uses the interrupt handler after the halt is cancelled.

(Note) Please don't switch the $\overline{HALT}$ signal to "Low" when the CPU executes the WAI instruction and is in the interrupt wait state to avoid the trouble of the CPU's operation after the halt is cancelled.

- **Bus Available (BA)**

This is an output control signal which is normally "Low" but "High" when the CPU accepts $\overline{HALT}$ and releases the buses. The HD6800 and HD6802 make BA "High" and release the buses at WAI execution, while the HD6303X doesn't make BA "High" under the same condition. But if the $\overline{HALT}$ becomes "Low" when the CPU is in the interrupt wait state after having executed the WAI, the CPU makes BA "High" and releases the buses. And when the $\overline{HALT}$ becomes "High", the CPU returns to the interrupt wait state.

■ PORT

The HD6303X provides three I/O ports. Table 2 gives the address of ports and the data direction register and Fig. 14 the block diagrams of each port.

Table 2 Port and Data Direction Register Address

Port	Port Address	Data Direction Register
Port 2	$0003	$0001
Port 5	$0015	–
Port 6	$0017	$0016

• Port 2

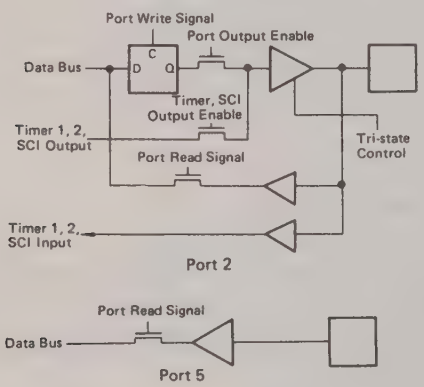
An 8-bit input/output port. The data direction register (DDR) of port 2 controls the I/O state. It provides two bits; bit 0 decides the I/O direction of P_{20} and bit 1 the I/O direction of P_{21} to P_{27} ("0" for input, "1" for output).

Port 2 is also used as an I/O pin for the timers and the SCI. When used as an I/O pin for the timers and the SCI, port 2 except P_{20} automatically becomes an input or an output depending on their functions regardless of the data direction register's value.

Port 2 Data Direction Register

7	6	5	4	3	2	1	0	
–	–	–	–	–	–	DDR 1 - 7	DDR 0	$0001

A reset clears the DDR of port 2 and configures port 2 as an input port. This port can drive one TTL and 30pF capacitance. In addition, it can produce 1mA current when V_{out} = 1.5V to drive directly the base of Darlington transistors.



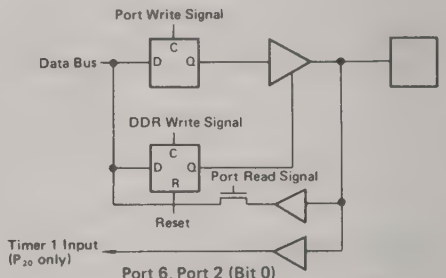


Figure 14 Port Block Diagram

• Port 5

An 8-bit port for input only. The lower four bits are also usable as input pins for interrupt, MR and $\overline{HALT}$.

• Port 6

An 8-bit I/O port. This port provides an 8-bit DDR corresponding to each bit and can specify input or output by the bit ("0" for input, "1" for output). This port can drive one TTL load and 30pF capacitance. A reset clears the DDR of port 6. In addition, it can produce 1mA current when V_{out} = 1.5V to drive directly the base of Darlington transistors.

■ BUS

• D_0~D_7

These pins are data bus and can drive one TTL load and 90pF capacitance respectively.

• A_0~A_{15}

These pins are address bus and can drive one TTL load and 90pF capacitance respectively.

■ RAM/PORT 5 CONTROL REGISTER

The control register located at $0014 controls on-chip RAM and port 5.

RAM/Port 5 Control Register

7	6	5	4	3	2	1	0	
STBY PWR	RAME	–	–	HLTE	MRE	IRQ_2 E	IRQ_1 E	$0014

Bit 0, Bit 1 $\overline{IRQ_1}$, $\overline{IRQ_2}$ Enable Bit (IRQ_1E, IRQ_2E)

When using P_{50} and P_{51} as interrupt pins, write "1" in these bits. When "0", the CPU doesn't accept an external interrupt or a sleep cancellation by the external interrupt. These bits become "0" during reset.

Bit 2 Memory Ready Enable Bit (MRE)

When using P_{52} as an input for Memory Ready signal, write "1" in this bit. When "0", the memory ready function is prohibited and P_{52} can be used as I/O port. This bit becomes "1" during reset.

Bit 3 Halt Enable bit (HLTE)

When using P_{53} as an input for Halt signal, write "1" in this

bit. When "0", the halt function is prohibited and P_{53} can be used as I/O port. This bit becomes "1" during reset.

(Note) When using P_{52} and P_{53} as the input ports in mode 1 and 2, MRE and HLTE bit should be cleared just after the reset.
Notice that memory ready and halt function is enable till MRE and HLTE bit is cleared.

Bit 4, Bit 5 Not Used.

Bit 6 RAM Enable (RAME)

On-chip RAM can be disabled by this control bit. By resetting the MPU, "1" is set to this bit, and on-chip RAM is enabled. This bit can be written "1" or "0" by software. When RAM is in disable condition (= logic "0"), on-chip RAM is invalid and the CPU can read data from external memory. This bit should be "0" before getting into the standby mode to protect on-chip RAM data.

Bit 7 Standby Power Bit (STBY PWR)

When V_{CC} is not provided in standby mode, this bit is cleared. This is a flag for both read/write by software. If this bit is set before standby mode, and remains set even after returning from standby mode, V_{CC} voltage is provided during standby mode and the on-chip RAM data is valid.

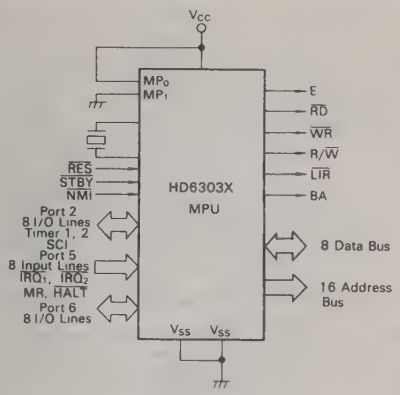


Figure 15 Operation Mode

■ MEMORY MAP

The MPU can address up to 65k bytes. Fig. 16 gives memory map of HD6303X. 32 internal registers use addresses from "00" as shown in Table 3.

Table 3 Internal Register

Address	Registers	R/W***	Initialize at RESET
00	—	—	—
01	Port 2 Data Direction Register	W	$FC
02*	—	—	—
03	Port 2	R/W	Undefined
04*	—	—	—
05	—	—	—
06*	—	—	—
07*	—	—	—
08	Timer Control/Status Register 1	R/W	$00
09	Free Running Counter ("High")	R/W	$00
0A	Free Running Counter ("Low")	R/W	$00
0B	Output Compare Register 1 ("High")	R/W	$FF
0C	Output Compare Register 1 ("Low")	R/W	$FF
0D	Input Capture Register ("High")	R	$00
0E	Input Capture Register ("Low")	R	$00
0F	Timer Control/Status Register 2	R/W	$10
10	Rate, Mode Control Register	R/W	$00
11	Tx/Rx Control Status Register	R/W	$20
12	Receive Data Register	R	$00
13	Transmit Data Register	W	$00
14	RAM/Port 5 Control Register	R/W	$7C or $FC
15	Port 5	R	—
16	Port 6 Data Direction Register	W	$00

(continued)

Table 3 Internal Register

Address	Registers	R/W***	Initialize at RESET
17	Port 6	R/W	Undefined
18*	—	—	—
19	Output Compare Register 2 ("High")	R/W	$FF
1A	Output Compare Register 2 ("Low")	R/W	$FF
1B	Timer Control/Status Register 3	R/W	$20
1C	Time Constant Register	W	$FF
1D	Timer 2 Up Counter	R/W	$00
1E	—	—	—
1F**	Test Register	—	—

* External Address.
** Test Register. Do not access to this register.
*** R : Read Only Register
W : Write Only Register
R/W : Read/Write Register

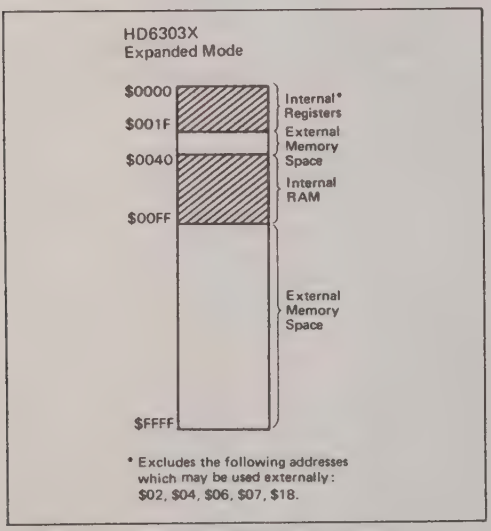


Figure 16 HD6303X Memory Map

■ TIMER 1

The HD6303X provides a 16-bit programmable timer which can simultaneously measure an input waveform and generate two independent output waveforms. The pulse widths of both input/output waveforms vary from microseconds to seconds.

Timer 1 is configurated as follows (refer to Fig. 18).

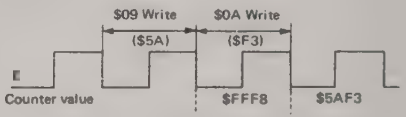
- Control/Status Register 1 (8 bit)
- Control/Status Register 2 (7 bit)
- Free Running Counter (16 bit)
- Output Compare Register 1 (16 bit)
- Output Compare Register 2 (16 bit)
- Input Capture Register (16 bit)

● Free-Running Counter (FRC) ($0009 : 000A)

The key timer element is a 16-bit free-running counter driven and incremented by system clock. The counter value is readable by software without affecting the counter. The counter is cleared by reset.

When writing to the upper byte ($09), the CPU writes the preset value ($FFF8) into the counter (address $09, $0A) regardless of the write data value. But when writing to the lower byte ($0A) after the upper byte writing, the CPU writes not only the lower byte data into lower 8 bit, but also the upper byte data into higher 8 bit of the FRC.

The counter will be as follows when the CPU writes to it by double store instructions (STD, STX etc.).



In the case of the CPU write ($5AF3) to the FRC

Figure 17 Counter Write Timing

● Output Compare Register (OCR) ($000B, $000C; OCR1) ($0019, $001A ; OCR2)

The output compare register is a 16-bit read/write register which can control an output waveform. The data of OCR is always compared with the FRC.

When the data matches, output compare flag (OCF) in the timer control/status register (TCSR) is set. If an output enable bit (OE) in the TCSR2 is "1", an output level bit (OLVL) in the TCSR will be output to bit 1 (Tout 1) and bit 5 (Tout 2) of port 2. To control the output level again by the next compare, the value of OCR and OLVL should be changed. The OCR is set to $FFFF at reset. The compare function is inhibited for a cycle just after a write to the OCR or to the upper byte of the FRC. This is to begin the comparison after setting the 16-bit value valid in the register and to inhibit the compare function at this cycle, because the CPU writes the upper byte to the FRC, and at the next cycle the counter is set to $FFF8.

* For data write to the FRC or the OCR, 2-byte transfer instruction (such as STX etc.) should be used.

● Input Capture Register (ICR) ($000D : 000E)

The input capture register is a 16-bit read only register which stores the FRC's value when external input signal transition

generates an input capture pulse. Such transition is controlled by input edge bit (IEDG) in the TCSR1.

In order to input the external input signal to the edge detecter, a bit of the DDR corresponding to bit 0 of port 2 should be cleared ("0"). When an input capture pulse occurs by the external input signal transition at the next cycle of CPU's high-byte read of the ICR, the input capture pulse will be delayed by one cycle. In order to ensure the input capture operation, a CPU read of the ICR needs 2-byte transfer instruction. The input pulse width should be at least 2 system cycles. This register is cleared ($0000) during reset.

- **Timer Control/Status Register 1 (TCSR1) ($0008)**

The timer control/status register 1 is an 8-bit register. All bits are readable and the lower 5 bits are also writable. The upper 3 bits are read only which indicate the following timer status.

Bit 5 The counter value reached to $0000 as a result of counting-up (TOF).

Bit 6 A match has occured between the FRC and the OCR 1 (OCF1).

Bit 7 Defined transition of the timer input signal causes the counter to transfer its data to the ICR (ICF).

The followings are each bit descriptions.

Timer Control/Status Register 1

7	6	5	4	3	2	1	0	
ICF	OCF1	TOF	EICI	EOCI1	ETOI	IEDG	OLVL1	$0008

Bit 0 OLVL1 Output Level 1

OLVL1 is transferred to port 2, bit 1 when a match occurs between the counter and the OCR1. If bit 0 of the TCSR2 (OE1) is set to "1", OLVL1 will appear at bit 1 of port 2.

Bit 1 IEDG Input Edge

This bit determines which edge, rising or falling, of input signal of port 2, bit 0 will trigger data transfer from the counter to the ICR. For this function, the DDR corresponding to port 2, bit 0 should be cleared beforehand.

IEDG=0, triggered on a falling edge ("High" to "Low")

IEDG=1, triggered on a rising edge ("Low" to "High")

Bit 2 ETOI Enable Timer Overflow Interrupt

When this bit is set, an internal interrupt (IRQ_3) by TOI interrupt is enabled. When cleared, the interrupt is inhibited.

Bit 3 EOCI1 Enable Output Compare Interrupt 1

When this bit is set, an internal interrupt (IRQ_3) by OCI1 interrupt is enabled. When cleared, the interrupt is inhibited.

Bit 4 EICI Enable Input Capture Interrupt

When this bit is set, an internal interrupt (IRQ_3) by ICI interrupt is enabled. When cleared, the interrupt is inhibited.

Bit 5 TOF Timer Overflow Flag

This read-only bit is set when the counter increments from $FFFF by 1. Cleared when the counter's upper byte ($0009) is ready by the CPU after the TCSR1 read.

Bit 6 OCF1 Output Compare Flag 1

This read-only bit is set when a match occurs between the OCR1 and the FRC. Cleared when writing to the OCR1 ($000B or $000C) after the TCSR1 or TCSR2 read.

Bit 7 ICF Input Capture Flag

This read-only bit is set when an input signal of port 2, bit 0 makes a transition as defined by IEDG and the FRC is transferred to the ICR. Cleared when reading the upper byte ($000D) of the ICR following the TCSR1 or TCSR2 read.

- **Timer Control/Status Register 2 (TCSR2) ($000F)**

The timer control/status register 2 is a 7-bit register. All bits are readable and the lower 4 bits are also writable. But the upper 3 bits are read-only which indicate the following timer status.

Bit 5 A match has occured between the FRC and the OCR2 (OCF2).

Bit 6 The same status flag as the OCF1 flag of the TCSR1, bit 6.

Bit 7 The same status flag as the ICF flag of the TCSR1, bit 7.

The followings are the each bit descriptions.

Timer Control/Status Register 2

7	6	5	4	3	2	1	0	
ICF	OCF1	OCF2	–	EOCI2	OLVL2	OE2	OE1	$000F

Bit 0 OE1 Output Enable 1

This bit enables the OLVL1 to appear at port 2, bit 1 when a match has occurred between the counter and the output compare register 1. When this bit is cleared, bit 1 of port 2 will be an I/O port. When set, it will be an output of OLVL1 automatically.

Bit 1 OE2 Output Enable 2

This bit enables the OLVL2 to appear at port 2, bit 5 when a match has occurred between the counter and the output compare register 2. When this bit is cleared, port 2, bit 5 will be an I/O port. When set, it will be an output of OLVL2 automatically.

Bit 2 OLVL2 Output Level 2

OLVL2 is transferred to port 2, bit 5 when a match has occurred between the counter and the OCR2. If bit 5 of the TCSR2 (OE2) is set to "1", OLVL2 will appear at port 2, bit 5.

Bit 3 EOCI2 Enable Output Compare Interrupt 2

When this bit is set, an internal interrupt (IRQ_3) by OCI2 interrupt is enabled. When cleared, the interrupt is inhibited.

Bit 4 Not Used

Bit 5 OCF2 Output Compare Flag 2

This read-only bit is set when a match has occurred between the counter and the OCR2. Cleared when writing to the OCR2 ($0019 or $001A) after the TCSR2 read.

Bit 6 OCF1 Output Compare Flag 1

Bit 7 ICF Input Capture Flag

OCF1 and ICF addresses are partially decoded. The CPU read of the TCSR1/TCSR2 makes it possible to read OCF1 and ICF into bit 6 and bit 7.

Both the TCSR1 and TCSR2 will be cleared during reset.

(Note) If OE1 or OE2 is set to "1" before the first output compare match occurs after reset restart, bit 1 or bit 5 of port 2 will produce "0" respectively.

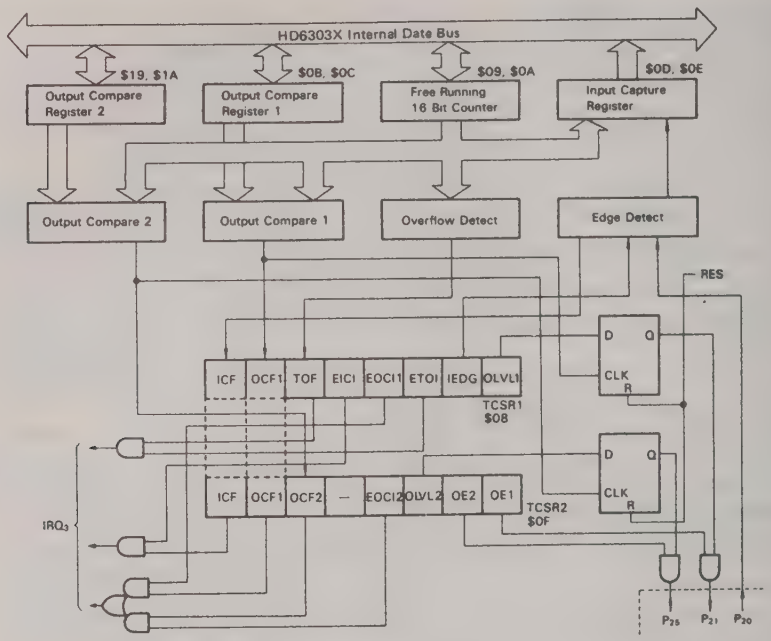


Figure 18 Timer 1 Block Diagram

■ TIMER 2

In addition to the timer 1, the HD6303X provides an 8-bit reloadable timer, which is capable of counting the external event. This timer 2 contains a timer output, so the MPU can generate three independent waveforms (refer to Fig. 19).

The timer 2 is configured as follows:

Control/Status Register 3 (7 bit)
8-bit Up Counter
Time Constant Register (8 bit)

• Timer 2 Up Counter (T2CNT) ($001D)

This is an 8-bit up counter which operates with the clock decided by CKS0 and CKS1 of the TCSR3. The CPU can read the value of the counter without affecting the counter. In addition, any value can be written to the counter by software even during counting.

The counter is cleared when a match occurs between the counter and the TCONR or during reset.

If a write operation is made by software to the counter at the cycle of counter clear, it does not reset the counter but put the write data to the counter.

• Time Constant Register (TCONR) ($001C)

The time constant register is an 8-bit write only register. It is always compared with the counter.

When a match has occurred, counter match flag (CMF) of the timer control status register 3 (TCSR3) is set and the value selected by TOS0 and TOS1 of the TCSR3 will appear at port 2, bit 6. When CMF is set, the counter will be cleared simultaneously and then start counting from $00. This enables regular interrupts and waveform outputs without any software support. The TCONR is set to "$FF" during reset.

• Timer Control/Status Register 3 (TCSR3) ($001B)

The timer control/status register 3 is a 7-bit register. All bits are readable and 6 bits except for CMF can be written.

The followings are each pin descriptions.

Timer Control/Status Register 3

7	6	5	4	3	2	1	0	
CMF	ECMI	—	T2E	TOS1	TOS0	CKS1	CKS0	$001B

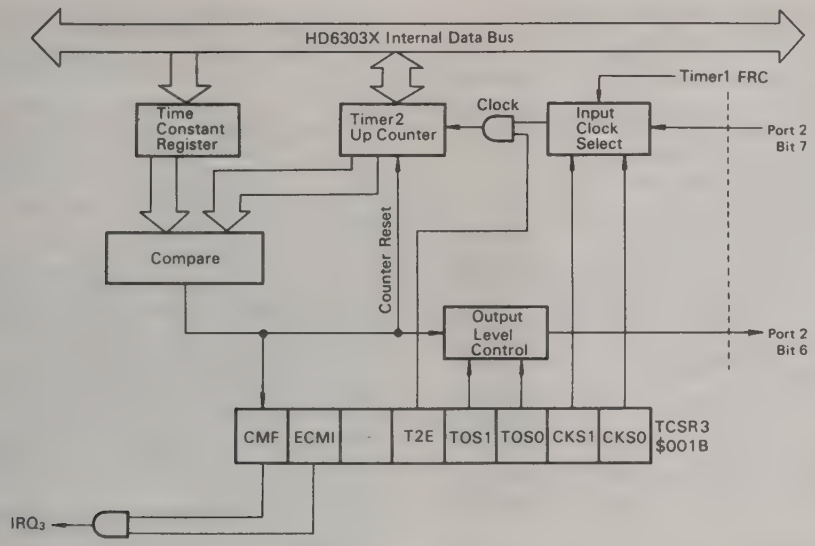


Figure 19 Timer 2 Block Diagram

Bit 0 CKS0 Input Clock Select 0
Bit 1 CKS1 Input Clock Select 1

Input clock to the counter is selected as shown in Table 4 depending on these two bits. When an external clock is selected, bit 7 of port 2 will be a clock input automatically. Timer 2 detects the rising edge of the external clock and increments the counter. The external clock is countable up to half the frequency of the system clock.

Table 4 Input Clock Select

CKS1	CKS0	Input Clock to the Counter
0	0	E clock
0	1	E clock/8*
1	0	E clock/128*
1	1	External clock

* These clocks come from the FRC of the timer 1. If one of these clocks is selected as an input clock to the up counter, the CPU should not write to the FRC of the timer 1.

Bit 2 TOS0 Timer Output Select 0
Bit 3 TOS1 Timer Output Select 1

When a match occurs between the counter and the TCONR timer 2 outputs shown in Table 5 will appear at port 2, bit 6 depending on these two bits. When both TOS0 and TOS1 are "0", bit 6 of port 2 will be an I/O port.

Table 5 Timer 2 Output Select

TOS1	TOS0	Timer Output
0	0	Timer Output Inhibited
0	1	Toggle Output*
1	0	Output "0"
1	1	Output "1"

* When a match occurs between the counter and the TCONR, timer 2 output level is reversed. This leads to production of a square wave with 50% duty to the external without any software support.

Bit 4 T2E Timer 2 Enable Bit

When this bit is cleared, a clock input to the up counter is prohibited and the up counter stops. When set to "1", a clock selected by CKS1 and CKS0 (Table 4) is input to the up counter.

(Note) P_{26} outputs "0" when T2E bit cleared and timer 2 set in output enable condition by TOS1 or TOS0. It also outputs "0" when T2E bit set "1" and timer 2 set in output enable condition before the first counter match occurs.

Bit 5 Not Used
Bit 6 ECMI Enable Counter Match Interrupt

When this bit is set, an internal interrupt (IRQ_3) by CMI is enabled. When cleared, the interrupt is inhibited.

Bit 7 CMF Counter Match Flag

This read-only bit is set when a match occurs between the up counter and the TCONR. Cleared by writing "0" by software write (unable to write "1" by software).

Each bit of the TCSR3 is cleared during reset.

■ SERIAL COMMUNICATION INTERFACE (SCI)

The HD6303X SCI contains two operation modes; one is an asynchronous mode by the NRZ format and the other is a clocked synchronous mode which transfers data synchronizing with the serial clock.

The SCI consists of the following registers as shown in Fig. 20 Block Diagram:

- Control/Status Register (TRCSR)
- Rate/Mode Control Register (RMCR)
- Receive Data Register (RDR)
- Receive Data Shift Register (RDSR)
- Transmit Data Register (TDR)
- Transmit Data Shift Register (TDSR)

The serial I/O hardware requires an initialization by software for operation. The procedure is usually as follows:

1) Write a desirable operation mode into each corresponding control bit of the RMCR.
2) Write a desirable operation mode into each corresponding control bit of the TRCSR.

When using bit 3 and 4 of port 2 for serial I/O only, there is no problem even if TE and RE bit are set. But when setting the baud rate and operation mode, TE and RE should be "0". When clearing TE and RE bit and setting them again, more than 1 bit cycle of the current baud rate is necessary. If set in less than 1 bit cycle, there may be a case that the internal transmit/receive initialization fails.

● Asynchronous Mode

An asynchronous mode contains the following two data formats:

1 Start Bit + 8 Bit Data + 1 Stop Bit
1 Start Bit + 9 Bit Data + 1 Stop Bit

In addition, if the 9th bit is set to "1" when making 9 bit data format, the format of

1 Start bit + 8 Bit Data + 2 Stop Bit

is also transferred.

Data transmission is enabled by setting TE bit of the TRCSR, then port 2, bit 4 will become a serial output independently of the corresponding DDR.

For data transmit, both the RMCR and TRCSR should be set under the desirable operating conditions. When TE bit is set during this process, 10 bit preamble will be sent in 8-bit data format and 11 bit in 9-bit data format. When the preamble is produced, the internal synchronization will become stable and the transmitter is ready to act.

The conditions at this stage are as follows.

1) If the TDR is empty (TDRE=1), consecutive 1's are produced to indicate the idle state.
2) If the TDR contains data (TDRE=0), data is sent to the transmit data shift register and data transmit starts.

During data transmit, a start bit of "0" is transmitted first. Then 8-bit or 9-bit data (starts from bit 0) and a stop bit "1" are transmitted.

When the TDR is "empty", hardware sets TDRE flag bit. If the CPU doesn't respond to the flag in proper timing (the TDRE is in set condition till the next normal data transfer starts from the transmit data register to the transmit sift register), "1" is transferred instead of the start bit "0" and continues to be transferred till data is provided to the data register. While the TDRE is "1", "0" is not transferred.

Data receive is possible by setting RE bit. This makes port 2, bit 3 be a serial input. The operation mode of data receive is decided by the contents of the TRCSR and RMCR. The first "0" (space) synchronizes the receive bit flow. Each bit of the following data will be strobed in the middle. If a stop bit is not "1", a framing error assumed and ORFE is set

When a framing error occurs, receive data is transferred to the receive data register and the CPU can read error-generating data. This makes it possible to detect a line break.

If the stop bit is "1", data is transferred to the receive data register and an interrupt flag RDRF is set. If RDRF is still set when receiving the stop bit of the next data, ORFE is set to indicate overrun generation.

When the CPU read the receive data register as a response to RDRF flag or ORFE flag after having read TRCS, RDRF or ORFE is cleared.

(Note) Clock Source in Asynchronous Mode

If CC1 : CC0 = 10, the internal bit rate clock is provided at P_{22} regardless of the values for TE or RE. Maximum clock rate is E÷16.

If both CC1 and CC0 are set, an external TTL compatible clock must be connected to P_{22} at sixteen times (16×) the desired bit rate, but not greater than E.

● Clocked Synchronous Mode

In the clocked synchronous mode, data transmit is synchronized with the clock pulse. The HD6303X SCI provides functionally independent transmitter and receiver which makes full duplex operation possible in the asynchronous mode. But in the clocked synchronous mode an SCI clock I/O pin is only P_{22}, so the simultaneous receive and transmit operation is not available. In this mode, TE and RE should not be in set condition ("1") simultaneously. Fig. 21 gives a synchronous clock and a data format in the clocked synchronous mode.

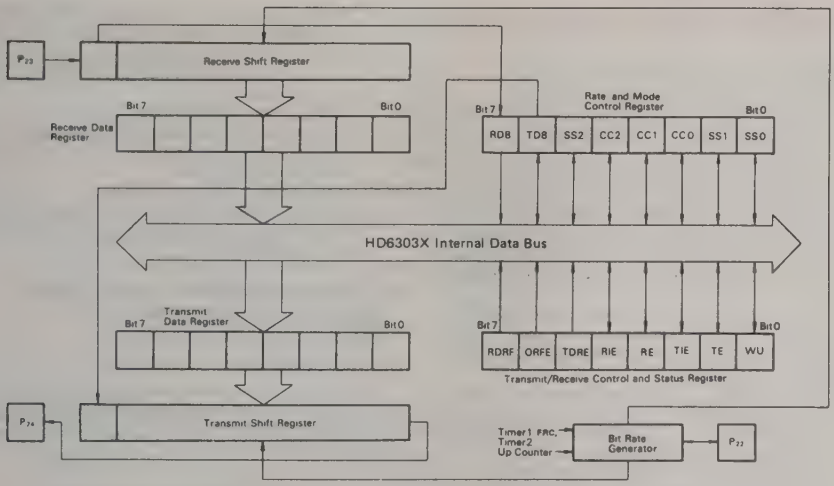


Figure 20 Serial Communication Interface Block Diagram

Data transmit is realized by setting TE bit in the TRCSR. Port 2, bit 4 becomes an output unconditionally independent of the value of the corresponding DDR.

Both the RMCR and TRCSR should be set in the desirable operating condition for data transmit.

When an external clock input is selected, data transmit is performed under the TDRE flag "0" from port 2, bit 4, synchronizing with 8 clock pulses input from external to port 2, bit 2.

Data is transmitted from bit 0 and the TDRE is set when the transmit data shift register is "empty". More than 9th clock pulse of external are ignored.

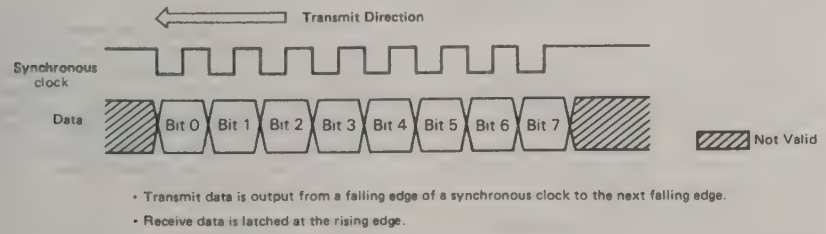


Figure 21 Clocked Synchronous Mode Format

When data transmit is selected to the clock output, the MPU produces transmit data and synchronous clock at TDRE flag clear.

Data receive is enabled by setting RE bit. Port 2, bit 3 will be a serial input. The operating mode of data receive is decided by the TRCSR and the RMCR.

If the external clock input is selected, RE bit should be set when P_{22} is "High". Then 8 external clock pulses and the synchronized receive data are input to port 2, bit 2 and bit 3 respectively. The MPU put receive data into the receive data shift register by this clock and set the RDRF flag at the termination of 8 bit data receive. More than 9th clock pulse of external input are ignored. When RDRF is cleared by reading the receive data register, the MPU starts receiving the next data. So RDRF should be cleared with P_{22} "High".

When data receive is selected to the clock output, 8 synchronous clocks are output to the external by setting RE bit. So receive data should be input from external, synchronously with this clock. When the first byte data is received, the RDRF flag is set. After the second byte, receive operation is performed and output the synchronous clock to the external by clearing the RDRF bit.

- **Transmit/Receive Control Status Register (TRCSR) ($0011)**

The TRCSR is composed of 8 bits which are all readable. Bits 0 to 4 are also writable. This register is initialized to $20 during reset. Each bit functions as follows.

Transmit/Receive Control Status Register

7	6	5	4	3	2	1	0	
RDRF	ORFE	TDRE	RIE	RE	TIE	TE	WU	$0011

Bit 0 WU Wake-up

In a typical multi-processor configuration, the software protocol provides the destination address at the first byte of the message. In order to make uninterested MPU ignore the remaining message, a wake-up function is available. By this, uninterested MPU can inhibit all further receive processing till the next message starts.

Then wake-up function is triggered by consecutive 1's with 1 frame length (10 bits for 8-bit data, 11 for 9-bit). The software protocol should provide the idle time between messages.

By setting this bit, the MPU stops data receive till the next message. The receive of consecutive "1" with one frame length wakes up and clears this bit and then the MPU restarts receive operation. However, the RE flag should be already set before setting this bit. In the clocked synchronous mode WU is not available, so this bit should not be set.

Bit 1 TE Transmit Enable

When this bit is set, transmit data will appear at port 2, bit 4 after one frame preamble in asynchronous mode, while in clocked synchronous mode it appears immediately. This is executed regardless of the value of the corresponding DDR. When TE is cleared, the serial I/O doesn't affect port 2, bit 4.

Bit 2 TIE Transmit Interrupt Enable

When this bit is set, an internal interrupt (IRQ_3) is enabled when TDRE (bit 5) is set. When cleared, the interrupt is inhibited.

Bit 3 RE Receive Enable

When set, a signal is input to the receiver from port 2, bit 3 regardless of the value of the DDR. When RE is cleared, the serial I/O doesn't affect port 2, bit 3.

Bit 4 RIE Receive Interrupt Enable

When this bit is set, an internal interrupt, IRQ_3 is enabled when RDRF (bit 7) or ORFE (bit 6) is set. When cleared, the interrupt is inhibited.

Bit 5 TDRE Transmit Data Register Empty

TDRE is set when the TDR is transferred to the transmit data shift register in the asynchronous mode, while in clocked synchronous mode when the TDSR is "empty". This bit is reset by reading the TRCSR and writing new transmit data to the transmit data register. TDRE is set to "1" during reset.

(Note) TDRE should be cleared in the transmittable state after the TE set.

Bit 6 ORFE Overrun Framing Error

ORFE is set by hardware when an overrun or a framing error is generated (during data receive only). An overrun error occurs when new receive data is ready to be transferred to the RDR during RDRF still being set. A framing error occurs when a stop bit is "0". But in clocked synchronous mode, this bit is not affected. This bit is cleared when reading the TRCSR, then the RDR, or during reset.

Bit 7 RDRF Receive Data Register Full

RDRF is set by hardware when the RDSR is transferred to the RDR. Cleared when reading the TRCSR, then the RDR, or during reset.

(Note) When a few bits are set between bit 5 to bit 7 in the TRCSR, a read of the TRCSR is sufficient for clearing those bits. It is not necessary to read the TRCSR every-time to clear each bit.

- **Transmit Rate/Mode Control Register (RMCR)**

The RMCR controls the following serial I/O:

- Baud Rate
- Clock Source
- Data Format
- Port 2, Bit 2 Function

In addition, if 9-bit data format is set in the asynchronous mode, the 9th bit is put in this register. All bits are readable and writable except bit 7 (read only). This register is set to $00 during reset.

Transfer Rate/Mode Control Register

7	6	5	4	3	2	1	0	
RD8	TD8	SS2	CC2	CC1	CC0	SS1	SS0	$0010

Bit 0	SS0	Speed Select
Bit 1	SS1	
Bit 5	SS2	

These bits control the baud rate used for the SCI. Table 6 lists the available baud rates. The timer 1 FRC (SS2=0) and the timer 2 up counter (SS2=1) provide the internal clock to the SCI. When selecting the timer 2 as a baud rate source, it functions as a baud rate generator. The timer 2 generates the baud rate listed in Table 7 depending on the value of the TCONR.

(Note) When operating the SCI with internal clock, do not perform write operation to the timer/counter which is the clock source of the SCI.

Bit 2	CC0	Clock Control/Format Select*
Bit 3	CC1	
Bit 4	CC2	

These bits control the data format and the clock source (refer to Table 8).

* CC0, CC1 and CC2 are cleared during reset and the MPU goes to the clocked synchronous mode of the external clock operation. Then the MPU sets port 2, bit 2 into the clock input state. When using port 2, bit 2 as an output port, the DDR of port 2 should be set to "1" and CC1 and CC0 to "0" and "1" respectively.

Table 6 SCI Bit Times and Transfer Rates

(1) Asynchronous Mode

			XTAL	2.4576MHz	4.0MHz	4.9152MHz
SS2	SS1	SS0	E	614.4kHz	1.0MHz	1.2288MHz
0	0	0	E÷16	26μs/38400Baud	16μs/62500Baud	13μs/76800Baud
0	0	1	E÷128	208μs/4800Baud	128μs/7812.5Baud	104.2μs/9600Baud
0	1	0	E÷1024	1.67ms/600Baud	1.024ms/976.6Baud	833.3μs/1200Baud
0	1	1	E÷4096	6.67ms/150Baud	4.096ms/244.1Baud	3.333ms/300Baud
1	—	—	—	*	*	*

* When SS2 is "1", Timer 2 provides SCI clocks. The baud rate is shown as follows with the TCONR as N.

$$\text{Baud Rate} = \frac{f}{32\,(N+1)} \qquad \begin{pmatrix} f: \text{input clock frequency to the timer 2 counter} \\ N = 0 \sim 255 \end{pmatrix}$$

(2) Clocked Synchronous Mode *

			XTAL	4.0MHz	6.0MHz	8.0MHz
SS2	SS1	SS0	E	1.0MHz	1.5MHz	2.0MHz
0	0	0	E÷2	2μs/bit	1.33μs/bit	1μs/bit
0	0	1	E÷16	16μs/bit	10.7μs/bit	8μs/bit
0	1	0	E÷128	128μs/bit	85.3μs/bit	64μs/bit
0	1	1	E÷512	512μs/bit	341μs/bit	256μs/bit
1	—	—	—	**	**	**

* Bit rates in the case of internal clock operation. In the case of external clock operation, the external clock is operatable up to DC ~ 1/2 system clock.

** The bit rate is shown as follows with the TCONR as N.

$$\text{Bit Rate } (\mu s/\text{bit}) = \frac{4\,(N+1)}{f} \qquad \begin{pmatrix} f: \text{input clock frequency to the timer 2 counter} \\ N = 0 \sim 255 \end{pmatrix}$$

Table 7 Baud Rate and Time Constant Register Example

Baud Rate (Baud) \ XTAL	2.4576MHz	3.6864MHz	4.0MHz	4.9152MHz	8.0MHz
110	21*	32*	35*	43*	70*
150	127	191	207	255	51*
300	63	95	103	127	207
600	31	47	51	63	103
1200	15	23	25	31	51
2400	7	11	12	15	25
4800	3	5	—	7	12
9600	1	2	—	3	—
19200	0	—	—	1	—
38400	—	—	—	0	—

* E/8 clock is input to the timer 2 up counter and E clock otherwise.

Table 8 SCI Format and Clock Source Control

CC2	CC1	CC0	Format	Mode	Clock Source	Port 2, Bit 2	Port 2, Bit 3	Port 2, Bit 4
0	0	0	8-bit data	Clocked Synchronous	External	Input	When the TRCSR, RE bit is "1", bit 3 is used as a serial input.	
0	0	1	8-bit data	Asynchronous	Internal	Not Used**		
0	1	0	8-bit data	Asynchronous	Internal	Output*		
0	1	1	8-bit data	Asynchronous	External	Input		
1	0	0	8-bit data	Clocked Synchronous	Internal	Output	When the TRCSR, TE bit is "1", bit 4 is used as a serial output.	
1	0	1	9-bit data	Asynchronous	Internal	Not Used**		
1	1	0	9-bit data	Asynchronous	Internal	Output*		
1	1	1	9-bit data	Asynchronous	External	Input		

* Clock output regardless of the TRCSR, bit RE and TE.
** Not used for the SCI.

Bit 6 TD8 Transmit Data Bit 8

When selecting 9-bit data format in the asynchronous mode, this bit is transmitted as the 9th data. In transmitting 9-bit data, write the 9th data into this bit then write data to the receive data register.

Bit 7 RD8 Receive Data Bit 8

When selecting 9-bit data format in the asynchronous mode, this bit stores the 9th bit data. In receiving 9-bit data, read this bit then the receive data register.

■ TIMER, SCI STATUS FLAG

Table 9 shows the set and reset conditions of each status flag in the timer 1, timer 2 and SCI.

As for Timer 1 and Timer 2 status flag, if the set and reset condition occur simultaneously, the set condition is prior to the reset condition. But in case of SCI control status flag, the reset condition has priority. Especially as for OCF1 and OCF2 of Timer 1, the set signal is generated periodically whenever FRC matches OCR after the set, and which can cause the unclear of the flag. To clear surely, the method is necessary to avoid the occurence of the set signal between TCSR Read and OCR write. For example, match the OCR value to FRC first, and next read TCSR, and then write OCR at once.

Table 9 Timer 1, Timer 2 and SCI Status Flag

		Set Condition	Reset Condition
Timer 1	ICF	FRC → ICR by edge input to P_{20}.	1. Read the TCSR1 or TCSR2 then ICRH, when ICF = 1 2. $\overline{RES}$=0
	OCF1	OCR1=FRC	1. Read the TCSR1 or TCSR2 then write to the OCR1H or OCR1L, when OCF1 = 1 2. $\overline{RES}$=0
	OCF2	OCR2=FRC	1. Read the TCSR2 then write to the OCR2H or OCR2L, when OCF2 = 1 2. $\overline{RES}$=0
	TOF	FRC=$FFFF+1 cycle	1. Read the TCSR1 then FRCH, when TOF = 1 2. $\overline{RES}$=0
Timer 2	CMF	T2CNT=TCONR	1. Write "0" to CMF, when CMF = 1 2. $\overline{RES}$=0
SCI	RDRF	Receive Shift Register → RDR	1. Read the TRCSR then RDR, when RDRF = 1 2. $\overline{RES}$=0
	ORFE	1. Framing Error (Asynchronous Mode) Stop Bit = 0 2. Overrun Error (Asynchronous Mode) Receive Shift Register → RDR when RDRF=1	1. Read the TRCSR then RDR, when ORFE = 1 2. $\overline{RES}$=0
	TDRE	1. Asynchronous Mode TDR → Transmit Shift Register 2. Clocked Synchronous Mode Transmit Shift Register is "empty" 3. $\overline{RES}$=0	Read the TRCSR then write to the TDR, when TDRE = 1 (Note) TDRE should be reset after the TE set.

(Note) 1. →; transfer
2. For example; "ICRH" means High byte of ICR.

■ LOW POWER DISSIPATION MODE

The HD6303X provides two low power dissipation modes; sleep and standby.

• Sleep Mode

The MPU goes to the sleep mode by SLP instruction execution. In the sleep mode, the CPU stops its operation, while the registers' contents are retained. In this mode, the peripherals except the CPU such as timers, SCI etc. continue their functions. The power dissipation of sleep-condition is one fifth that of operating condition.

The MPU returns from this mode by an interrupt, $\overline{RES}$ or $\overline{STBY}$; it goes to the reset state by $\overline{RES}$ and the standby mode by $\overline{STBY}$. When the CPU acknowledges an interrupt request, it cancels the sleep mode, returns to the operation mode and branches to the interrupt routine. When the CPU masks this interrupt, it cancels the sleep mode and executes the next instruction. However, for example if the timer 1 or 2 prohibits a timer interrupt, the CPU doesn't cancel the sleep mode because of no interrupt request.

This sleep mode is effective to reduce the power dissipation for a system with no need of the HD6303X's consecutive operation.

• Standby Mode

The HD6303X stops all the clocks and goes to the reset state with $\overline{STBY}$ "Low". In this mode, the power dissipation is reduced conspicuously. All pins except for the power supply, the $\overline{STBY}$ and XTAL are detached from the MPU internally and go to the high impedance state.

In this mode the power is supplied to the HD6303X, so the contents of RAM is retained. The MPU returns from this mode during reset. The followings are typical usage of this mode.

Save the CPU information and SP contents on RAM by $\overline{NMI}$. Then disable the RAME bit of the RAM control register and set the STBY PWR bit to go to the standby mode. If the STBY PWR bit is still set at reset start, that indicates the power is supplied to the MPU and RAM contents are retained properly. So system can restore itself by returning their pre-standby informations to the SP and the CPU. Fig. 22 depicts the timing at each pin with this example.

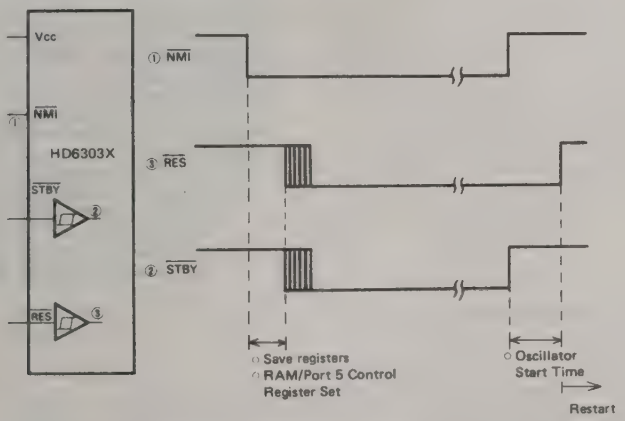


Figure 22 Standby Mode Timing

■ TRAP FUNCTION

The CPU generates an interrupt with the highest priority (TRAP) when fetching an undefined instruction or an instruction from non-memory space. The TRAP prevents the system-burst caused by noise or a program error.

• Op Code Error

When fetching an undefined op code, the CPU saves CPU registers as well as a normal interrupt and branches to the TRAP ($FFEE, $FFEF). This has the priority next to reset.

• Address Error

When an instruction fetch is made from internal register ($0000~$001F), the MPU generates an interrupt as well as an op code error. But on the system with no memory in its external memory area, this function is not applicable if an instruction fetch is made from the external non-memory area.

This function is available only for an instruction fetch and is not applicable to the access of normal data read/write.

(Note) The TRAP interrupt provides a retry function differently from other interrupts. This is a program flow return to the address where the TRAP occurs when a sequence returns to a main routine from the TRAP interrupt routine by RTI. The retry can prevent the system burst caused by noise etc.
However, if another TRAP occurs, the program repeats the TRAP interrupt forever, so the consideration is necessary in programming.

■ INSTRUCTION SET

The HD6303X provides object code upward compatible with the HD6801 to utilize all instruction' set of the HMCS6800. It also reduces the execution times of key instruc-

tions for throughput improvement.

Bit manipulation instruction, change instruction of the index register and accumulator and sleep instruction are also added.

The followings are explained here.

- CPU Programming Model (refer to Fig. 23)
- Addressing Mode
- Accumulator and Memory Manipulation Instruction (refer to Table 10)
- New Instruction
- Index Register and Stack Manipulation Instruction (refer to Table 11)
- Jump and Branch Instruction (refer to Table 12)
- Condition Code Register Manipulation (refer to Table 13)
- Op Code Map (refer to Table 14)

• Programming Model

Fig. 23 depicts the HD6303X programming model. The double accumulator D consists of accumulator A and B, so when using the accumulator D, the contents of A and B are destroyed.

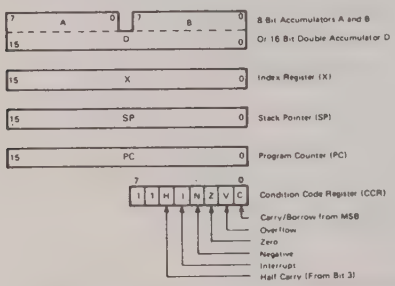


Figure 23 CPU Programming Model

• CPU Addressing Mode

The HD6303X provides 7 addressing modes. The addressing mode is decided by an instruction type and code. Table 10 through 14 show addressing modes of each instruction with the execution times counted by the machine cycle.

When the clock frequency is 4 MHz, the machine cycle time becomes microseconds directly.

Accumulator (ACCX) Addressing

Only an accumulator is addressed and the accumulator A or B is selected. This is a one-byte instruction.

Immediate Addressing

This addressing locates a data in the second byte of an instruction. However, LDS and LDX locate a data in the second and third byte exceptionally. This addressing is a 2 or 3-byte instruction.

Direct Addressing

In this addressing mode, the second byte of an instruction shows the address where a data is stored. 256 bytes ($0 through $255) can be addressed directly. Execution times can be reduced by storing data in this area so it is recommended to make it RAM for users' data area in configurating a system. This is a 2-byte instruction, while 3-byte with regard to AIM, OIM, EIM and TIM.

Extended Addressing

In this mode, the second byte shows the upper 8 bit of the data stored address and the third byte the lower 8 bit. This indicates the absolute address of 3-byte instruction in the memory.

Indexed Addressing

The second byte of an instruction and the lower 8 bit of the index register are added in this mode. As for AIM, OIM, EIM and TIM, the third byte of an instruction and the lower 8 bits of the index register are added.

This carry is added to the upper 8 bit of the index register and the result is used for addressing the memory. The modified address is retained in the temporary address register, so the contents of the index register doesn't change. This is a 2-byte instruction except AIM, OIM, EIM and TIM (3-byte instruction).

Implied Addressing

An instruction itself specifies the address. That is, the instruction addresses a stack pointer, index register etc. This is a one-byte instruction.

Relative Addressing

The second byte of an instruction and the lower 8 bits of the program counter are added. The carry or borrow is added to the upper 8 bit. So addressing from −126 to +129 byte of the current instruction is enabled. This is a 2-byte instruction.

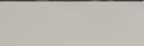
(Note) CLI, SEI Instructions and Interrupt Operation

When accepting the IRQ at a preset timing with CLI and SEI instructions, more than 2 cycles are necessary between the CLI and SEI instructions. For example, the following program (a) (b) don't accept the IRQ but (c) accepts it.

.	.	.
.	.	.
.	.	.
.	.	CLI
CLI	CLI	NOP
SEI	NOP	NOP
.	SEI	SEI
.	.	.
.	.	.
.	.	.
.	.	.
(a)	(b)	(c)

The same thing can be said to the TAP instruction instead of the CLI and SEI instructions.

Table 10 Accumulator, Memory Manipulation Instructions

Operations	Mnemonic	Addressing Modes															Boolean/ Arithmetic Operation	Condition Code Register					
		IMMED.			DIRECT			INDEX			EXTEND			IMPLIED				5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#		H	I	N	Z	V	C
Add	ADDA	8B	2	2	9B	3	2	AB	4	2	BB	4	3				A + M → A	↕	•	↕	↕	↕	↕
	ADDB	CB	2	2	DB	3	2	EB	4	2	FB	4	3				B + M → B	↕	•	↕	↕	↕	↕
Add Double	ADDD	C3	3	3	D3	4	2	E3	5	2	F3	5	3				A : B + M : M + 1 → A : B	•	•	↕	↕	↕	↕
Add Accumulators	ABA													1B	1	1	A + B → A	↕	•	↕	↕	↕	↕
Add With Carry	ADCA	89	2	2	99	3	2	A9	4	2	B9	4	3				A + M + C → A	↕	•	↕	↕	↕	↕
	ADCB	C9	2	2	D9	3	2	E9	4	2	F9	4	3				B + M + C → B	↕	•	↕	↕	↕	↕
AND	ANDA	84	2	2	94	3	2	A4	4	2	B4	4	3				A·M → A	•	•	↕	↕	R	•
	ANDB	C4	2	2	D4	3	2	E4	4	2	F4	4	3				B·M → B	•	•	↕	↕	R	•
Bit Test	BIT A	85	2	2	95	3	2	A5	4	2	B5	4	3				A·M	•	•	↕	↕	R	•
	BIT B	C5	2	2	D5	3	2	E5	4	2	F5	4	3				B·M	•	•	↕	↕	R	•
Clear	CLR							6F	5	2	7F	5	3				00 → M	•	•	R	S	R	R
	CLRA													4F	1	1	00 → A	•	•	R	S	R	R
	CLRB													5F	1	1	00 → B	•	•	R	S	R	R
Compare	CMPA	81	2	2	91	3	2	A1	4	2	B1	4	3				A − M	•	•	↕	↕	↕	↕
	CMPB	C1	2	2	D1	3	2	E1	4	2	F1	4	3				B − M	•	•	↕	↕	↕	↕
Compare Accumulators	CBA													11	1	1	A − B	•	•	↕	↕	↕	↕
Complement, 1's	COM							63	6	2	73	6	3				$\overline{M}$ → M	•	•	↕	↕	R	S
	COMA													43	1	1	$\overline{A}$ → A	•	•	↕	↕	R	S
	COMB													53	1	1	$\overline{B}$ → B	•	•	↕	↕	R	S
Complement, 2's (Negate)	NEG							60	6	2	70	6	3				00 − M → M	•	•	↕	↕	①	②
	NEGA													40	1	1	00 − A → A	•	•	↕	↕	①	②
	NEGB													50	1	1	00 − B → B	•	•	↕	↕	①	②
Decimal Adjust, A	DAA													19	2	1	Converts binary add of BCD characters into BCD format	•	•	↕	↕	↕	③
Decrement	DEC							6A	6	2	7A	6	3				M − 1 → M	•	•	↕	↕	④	•
	DECA													4A	1	1	A − 1 → A	•	•	↕	↕	④	•
	DECB													5A	1	1	B − 1 → B	•	•	↕	↕	④	•
Exclusive OR	EORA	88	2	2	98	3	2	A8	4	2	B8	4	3				A ⊕ M → A	•	•	↕	↕	R	•
	EORB	C8	2	2	D8	3	2	E8	4	2	F8	4	3				B ⊕ M → B	•	•	↕	↕	R	•
Increment	INC							6C	6	2	7C	6	3				M + 1 → M	•	•	↕	↕	⑤	•
	INCA													4C	1	1	A + 1 → A	•	•	↕	↕	⑤	•
	INCB													5C	1	1	B + 1 → B	•	•	↕	↕	⑤	•
Load Accumulator	LDAA	86	2	2	96	3	2	A6	4	2	B6	4	3				M → A	•	•	↕	↕	R	•
	LDAB	C6	2	2	D6	3	2	E6	4	2	F6	4	3				M → B	•	•	↕	↕	R	•
Load Double Accumulator	LDD	CC	3	3	DC	4	2	EC	5	2	FC	5	3				M + 1 → B, M → A	•	•	↕	↕	R	•
Multiply Unsigned	MUL													3D	7	1	A x B → A : B	•	•	•	•	•	⑪
OR, Inclusive	ORAA	8A	2	2	9A	3	2	AA	4	2	BA	4	3				A + M → A	•	•	↕	↕	R	•
	ORAB	CA	2	2	DA	3	2	EA	4	2	FA	4	3				B + M → B	•	•	↕	↕	R	•
Push Data	PSHA													36	4	1	A → Msp, SP − 1 → SP	•	•	•	•	•	•
	PSHB													37	4	1	B → Msp, SP − 1 → SP	•	•	•	•	•	•
Pull Data	PULA													32	3	1	SP + 1 → SP, Msp → A	•	•	•	•	•	•
	PULB													33	3	1	SP + 1 → SP, Msp → B	•	•	•	•	•	•
Rotate Left	ROL							69	6	2	79	6	3					•	•	↕	↕	⑥	↕
	ROLA													49	1	1		•	•	↕	↕	⑥	↕
	ROLB													59	1	1		•	•	↕	↕	⑥	↕
Rotate Right	ROR							66	6	2	76	6	3					•	•	↕	↕	⑥	↕
	RORA													46	1	1		•	•	↕	↕	⑥	↕
	RORB													56	1	1		•	•	↕	↕	⑥	↕

(Note) Condition Code Register will be explained in Note of Table 13.

(continued)

Table 10 Accumulator, Memory Manipulation Instructions

Operations	Mnemonic	IMMED. OP	IMMED. ~	IMMED. #	DIRECT OP	DIRECT ~	DIRECT #	INDEX OP	INDEX ~	INDEX #	EXTEND OP	EXTEND ~	EXTEND #	IMPLIED OP	IMPLIED ~	IMPLIED #	Boolean/ Arithmetic Operation	5 H	4 I	3 N	2 Z	1 V	0 C
Shift Left Arithmetic	ASL							68	6	2	78	6	3				M	•	•	↕	↕	⑤	↕
	ASLA													48	1	1	A	•	•	↕	↕	⑤	↕
	ASLB													58	1	1	B	•	•	↕	↕	⑥	↕
Double Shift Left, Arithmetic	ASLD													05	1	1	C b7 … b0 ← 0	•	•	↕	↕	⑥	↕
Shift Right Arithmetic	ASR							67	6	2	77	6	3				M	•	•	↕	↕	⑥	↕
	ASRA													47	1	1	A	•	•	↕	↕	⑤	↕
	ASRB													57	1	1	B	•	•	↕	↕	⑥	↕
Shift Right Logical	LSR							64	6	2	74	6	3				M	•	•	R	↕	⑤	↕
	LSRA													44	1	1	A	•	•	R	↕	⑥	↕
	LSRB													54	1	1	B	•	•	R	↕	⑥	↕
Double Shift Right Logical	LSRD													04	1	1	0 → ACC A/ ACC B → C	•	•	R	↕	⑥	↕
Store Accumulator	STAA				97	3	2	A7	4	2	B7	4	3				A → M	•	•	↕	↕	R	•
	STAB				D7	3	2	E7	4	2	F7	4	3				B → M	•	•	↕	↕	R	•
Store Double Accumulator	STD				DD	4	2	ED	5	2	FD	5	3				A → M B → M + 1	•	•	↕	↕	R	•
Subtract	SUBA	80	2	2	90	3	2	A0	4	2	B0	4	3				A − M → A	•	•	↕	↕	↕	↕
	SUBB	C0	2	2	D0	3	2	E0	4	2	F0	4	3				B − M → B	•	•	↕	↕	↕	↕
Double Subtract	SUBD	83	3	3	93	4	2	A3	5	2	B3	5	3				A B − M (M + 1) → A B	•	•	↕	↕	↕	↕
Subtract Accumulators	SBA													10	1	1	A − B → A	•	•	↕	↕	↕	↕
Subtract With Carry	SBCA	82	2	2	92	3	2	A2	4	2	B2	4	3				A − M − C → A	•	•	↕	↕	↕	↕
	SBCB	C2	2	2	D2	3	2	E2	4	2	F2	4	3				B − M − C → B	•	•	↕	↕	↕	↕
Transfer Accumulators	TAB													16	1	1	A → B	•	•	↕	↕	R	•
	TBA													17	1	1	B → A	•	•	↕	↕	R	•
Test Zero or Minus	TST							6D	4	2	7D	4	3				M − 00	•	•	↕	↕	R	R
	TSTA													4D	1	1	A − 00	•	•	↕	↕	R	R
	TSTB													5D	1	1	B − 00	•	•	↕	↕	R	R
And Immediate	AIM				71	6	3	61	7	3							M · IMM → M	•	•	↕	↕	R	•
OR Immediate	OIM				72	6	3	62	7	3							M + IMM → M	•	•	↕	↕	R	•
EOR Immediate	EIM				75	6	3	65	7	3							M ⊕ IMM → M	•	•	↕	↕	R	•
Test Immediate	TIM				7B	4	3	6B	5	3							M · IMM	•	•	↕	↕	R	•

(Note) Condition Code Register will be explained in Note of Table 13.

- **Additional Instruction**

In addition to the HD6801 instruction set, the HD6303X prepares the following new instructions.

AIM . . . . . (M)·(IMM) → (M)

Executes "AND" operation to immediate data and the memory contents and stores its result in the memory.

OIM . . . . . (M) + (IMM) → (M)

Executes "OR" operation to immediate data and the memory contents and stores its result in the memory.

EIM . . . . . (M) ⊕ (IMM) → (M)

Executes "EOR" operation to immediate data and the memory contents and stores its result in the memory.

TIM . . . . . (M) · (IMM)

Executes "AND" operation to immediate data and changes the relative flag of the condition code register.

These area 3-byte instructions; the first byte is op code, the second immediate data and the third address modifier.

XGDX . . . . . (ACCD) ↔ (IX)

Exchanges the contents of accumulator and the index register.

SLP

Goes to the sleep mode. Refer to "LOW POWER DISSIPATION MODE" for more details of the sleep mode.

Table 11 Index Register, Stack Manipulation Instructions

Pointer Operations	Mnemonic	Addressing Modes															Boolean/ Arithmetic Operation	Condition Code Register					
		IMMED.			DIRECT			INDEX			EXTEND			IMPLIED				5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#		H	I	N	Z	V	C
Compare Index Reg	CPX	8C	3	3	9C	4	2	AC	5	2	BC	5	3				X − M M+1	•	•	↕	↕	↕	↕
Decrement Index Reg	DEX													09	1	1	X − 1 → X	•	•	•	↕	•	•
Decrement Stack Pntr	DES													34	1	1	SP − 1 → SP	•	•	•	•	•	•
Increment Index Reg	INX													08	1	1	X + 1 → X	•	•	•	↕	•	•
Increment Stack Pntr	INS													31	1	1	SP + 1 → SP	•	•	•	•	•	•
Load Index Reg	LDX	CE	3	3	DE	4	2	EE	5	2	FE	5	3				M → X_H, (M + 1) → X_L	•	•	⑦	↕	R	•
Load Stack Pntr	LDS	8E	3	3	9E	4	2	AE	5	2	BE	5	3				M → SP_H, (M+1) → SP_L	•	•	⑦	↕	R	•
Store Index Reg	STX				DF	4	2	EF	5	2	FF	5	3				X_H → M, X_L → (M + 1)	•	•	⑦	↕	R	•
Store Stack Pntr	STS				9F	4	2	AF	5	2	BF	5	3				SP_H → M, SP_L → (M+1)	•	•	⑦	↕	R	•
Index Reg → Stack Pntr	TXS													35	1	1	X − 1 → SP	•	•	•	•	•	•
Stack Pntr → Index Reg	TSX													30	1	1	SP + 1 → X	•	•	•	•	•	•
Add	ABX													3A	1	1	B + X → X	•	•	•	•	•	•
Push Data	PSHX													3C	5	1	X_L → M_{SP}, SP − 1 → SP X_H → M_{SP}, SP − 1 → SP	•	•	•	•	•	•
Pull Data	PULX													38	4	1	SP + 1 → SP, M_{SP} → X_H SP + 1 → SP, M_{SP} → X_L	•	•	•	•	•	•
Exchange	XGDX													18	2	1	ACCD↔IX	•	•	•	•	•	•

(Note) Condition Code Register will be explained in Note of Table 13.

Table 12 Jump, Branch Instructions

Operations	Mnemonic	Addressing Modes															Branch Test	Condition Code Register					
		RELATIVE			DIRECT			INDEX			EXTEND			IMPLIED				5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#		H	I	N	Z	V	C
Branch Always	BRA	20	3	2													None	•	•	•	•	•	•
Branch Never	BRN	21	3	2													None	•	•	•	•	•	•
Branch If Carry Clear	BCC	24	3	2													C = 0	•	•	•	•	•	•
Branch If Carry Set	BCS	25	3	2													C = 1	•	•	•	•	•	•
Branch If = Zero	BEQ	27	3	2													Z = 1	•	•	•	•	•	•
Branch If ≥ Zero	BGE	2C	3	2													N ⊕ V = 0	•	•	•	•	•	•
Branch If > Zero	BGT	2E	3	2													Z + (N ⊕ V) = 0	•	•	•	•	•	•
Branch If Higher	BHI	22	3	2													C + Z = 0	•	•	•	•	•	•
Branch If ≤ Zero	BLE	2F	3	2													Z + (N ⊕ V) = 1	•	•	•	•	•	•
Branch If Lower Or Same	BLS	23	3	2													C + Z = 1	•	•	•	•	•	•
Branch If < Zero	BLT	2D	3	2													N ⊕ V = 1	•	•	•	•	•	•
Branch If Minus	BMI	2B	3	2													N = 1	•	•	•	•	•	•
Branch If Not Equal Zero	BNE	26	3	2													Z = 0	•	•	•	•	•	•
Branch If Overflow Clear	BVC	28	3	2													V = 0	•	•	•	•	•	•
Branch If Overflow Set	BVS	29	3	2													V = 1	•	•	•	•	•	•
Branch If Plus	BPL	2A	3	2													N = 0	•	•	•	•	•	•
Branch To Subroutine	BSR	8D	5	2														•	•	•	•	•	•
Jump	JMP							6E	3	2	7E	3	3					•	•	•	•	•	•
Jump To Subroutine	JSR				9D	5	2	AD	5	2	BD	6	3					•	•	•	•	•	•
No Operation	NOP													01	1	1	Advances Prog Cntr Only	•	•	•	•	•	•
Return From Interrupt	RTI													3B	10	1		———		⑧		———	
Return From Subroutine	RTS													39	5	1		•	•	•	•	•	•
Software Interrupt	SWI													3F	12	1		•	S	•	•	•	•
Wait for Interrupt*	WAI													3E	9	1		•	⑨	•	•	•	•
Sleep	SLP													1A	4	1		•	•	•	•	•	•

(Note) * WAI puts R/$\overline{W}$ high; Address Bus goes to FFFF; Data Bus goes to the three state.
Condition Code Register will be explained in Note of Table 13.

Table 13 Condition Code Register Manipulation Instructions

Operations	Mnemonic	Addressing Modes IMPLIED OP	~	#	Boolean Operation	Condition Code Register 5 H	4 I	3 N	2 Z	1 V	0 C
Clear Carry	CLC	0C	1	1	0 → C	•	•	•	•	•	R
Clear Interrupt Mask	CLI	0E	1	1	0 → I	•	R	•	•	•	•
Clear Overflow	CLV	0A	1	1	0 → V	•	•	•	•	R	•
Set Carry	SEC	0D	1	1	1 → C	•	•	•	•	•	S
Set Interrupt Mask	SEI	0F	1	1	1 → I	•	S	•	•	•	•
Set Overflow	SEV	0B	1	1	1 → V	•	•	•	•	S	•
Accumulator A → CCR	TAP	06	1	1	A → CCR	⑩					
CCR → Accumulator A	TPA	07	1	1	CCR → A	•	•	•	•	•	•

LEGEND

OP Operation Code (Hexadecimal)
~ Number of MCU Cycles
M_{SP} Contents of memory location pointed to by Stack Pointer
Number of Program Bytes
+ Arithmetic Plus
− Arithmetic Minus
• Boolean AND
+ Boolean Inclusive OR
⊕ Boolean Exclusive OR
$\overline{M}$ Complement of M
→ Transfer into
0 Bit = Zero
00 Byte = Zero

CONDITION CODE SYMBOLS

H Half-carry from bit 3 to bit 4
I Interrupt mask
N Negative (sign bit)
Z Zero (byte)
V Overflow, 2's complement
C Carry/Borrow from/to bit 7
R Reset Always
S Set Always
↕ Set if true after test or clear
• Not Affected

(Note) Condition Code Register Notes: (Bit set if test is true and cleared otherwise)

① (Bit V) Test: Result = 10000000?
② (Bit C) Test: Result ≠ 00000000?
③ (Bit C) Test: BCD Character of high-order byte greater than 10? (Not cleared if previously set)
④ (Bit V) Test: Operand = 10000000 prior to execution?
⑤ (Bit V) Test: Operand = 01111111 prior to execution?
⑥ (Bit V) Test: Set equal to N⊕ C = 1 after the execution of instructions
⑦ (Bit N) Test: Result less than zero? (Bit 15=1)
⑧ (All Bit) Load Condition Code Register from Stack.
⑨ (Bit I) Set when interrupt occurs. If previously set, a Non-Maskable Interrupt is required to exit the wait state.
⑩ (All Bit) Set according to the contents of Accumulator A.
⑪ (Bit C) Result of Multiplication Bit 7=1? (ACCB)

Table 14 OP-Code Map

OP CODE HI	LO	0000 0	0001 1	0010 2	0011 3	ACC A 0100 4	ACC B 0101 5	IND 0110 6	EXT / DIR* 0111 7	ACCA or SP IMM 1000 8	DIR 1001 9	IND 1010 A	EXT 1011 B	ACCB or X IMM 1100 C	DIR 1101 D	IND 1110 E	EXT 1111 F	HI / LO
0000	0		SBA	BRA	TSX	NEG							SUB					0
0001	1	NOP	CBA	BRN	INS			AIM					CMP					1
0010	2			BHI	PULA			OIM					SBC					2
0011	3			BLS	PULB	COM					SUBD				ADDD			3
0100	4	LSRD		BCC	DES	LSR							AND					4
0101	5	ASLD		BCS	TXS			EIM					BIT					5
0110	6	TAP	TAB	BNE	PSHA	ROR							LDA					6
0111	7	TPA	TBA	BEQ	PSHB	ASR					STA				STA			7
1000	8	INX	XGDX	BVC	PULX	ASL							EOR					8
1001	9	DEX	DAA	BVS	RTS	ROL							ADC					9
1010	A	CLV	SLP	BPL	ABX	DEC							ORA					A
1011	B	SEV	ABA	BMI	RTI			TIM					ADD					B
1100	C	CLC		BGE	PSHX	INC					CPX				LDD			C
1101	D	SEC		BLT	MUL	TST				BSR		JSR			STD			D
1110	E	CLI		BGT	WAI			JMP			LDS				LDX			E
1111	F	SEI		BLE	SWI	CLR						STS				STX		F
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	

*UNDEFINED OP CODE

* Only each instructions of AIM, OIM, EIM, TIM

■ CPU OPERATION

• CPU Instruction Flow

When operating, the CPU fetches an instruction from a memory and executes the required function. This sequence starts with $\overline{RES}$ cancel and repeats itself limitlessly if not affected by a special instruction or a control signal. SWI, RTI, WAI and SLP instructions change this operation, while $\overline{NMI}$, $\overline{IRQ_1}$, $\overline{IRQ_2}$, IRQ_3, $\overline{HALT}$ and $\overline{STBY}$ control it. Fig. 24 gives the CPU mode transition and Fig. 25 the CPU system flow chart. Table 15 shows CPU operating states and port states.

• Operation at Each Instruction Cycle

Table 16 shows the operation at each instruction cycle. By the pipeline control of the HD6303X, MULT, PUL, DAA and XGDX instructions etc. prefetch the next instruction. So attention is necessary to the counting of the instruction cycles because it is different from the usual one ------ op code fetch to the next instruction op code.

Table 15 CPU Operation State and Port State

Port	Reset	STBY***	HALT	Sleep
$A_0 \sim A_7$	H	T	T	H
Port 2	T	T	Keep	Keep
$D_0 \sim D_7$	T	T	T	T
$A_8 \sim A_{15}$	H	T	T	H
Port 5	T	T	T	T
Port 6	T	T	Keep	Keep
Control Signal	*	T	**	*

H ; High, L ; Low, T ; High Impedance
* $\overline{RD}$, $\overline{WR}$, R/$\overline{W}$, $\overline{LIR}$ = H, BA = L
** $\overline{RD}$, $\overline{WR}$, R/$\overline{W}$ = T, $\overline{LIR}$, BA = H
*** E pin goes to high impedance state.

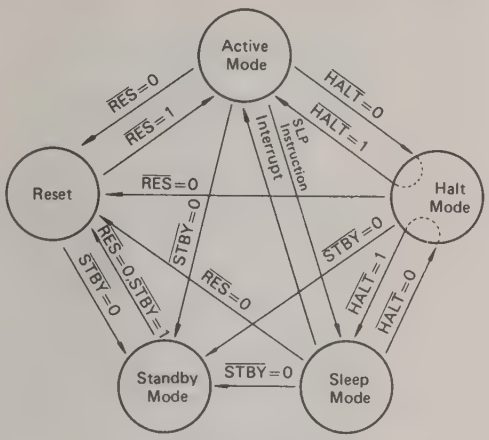
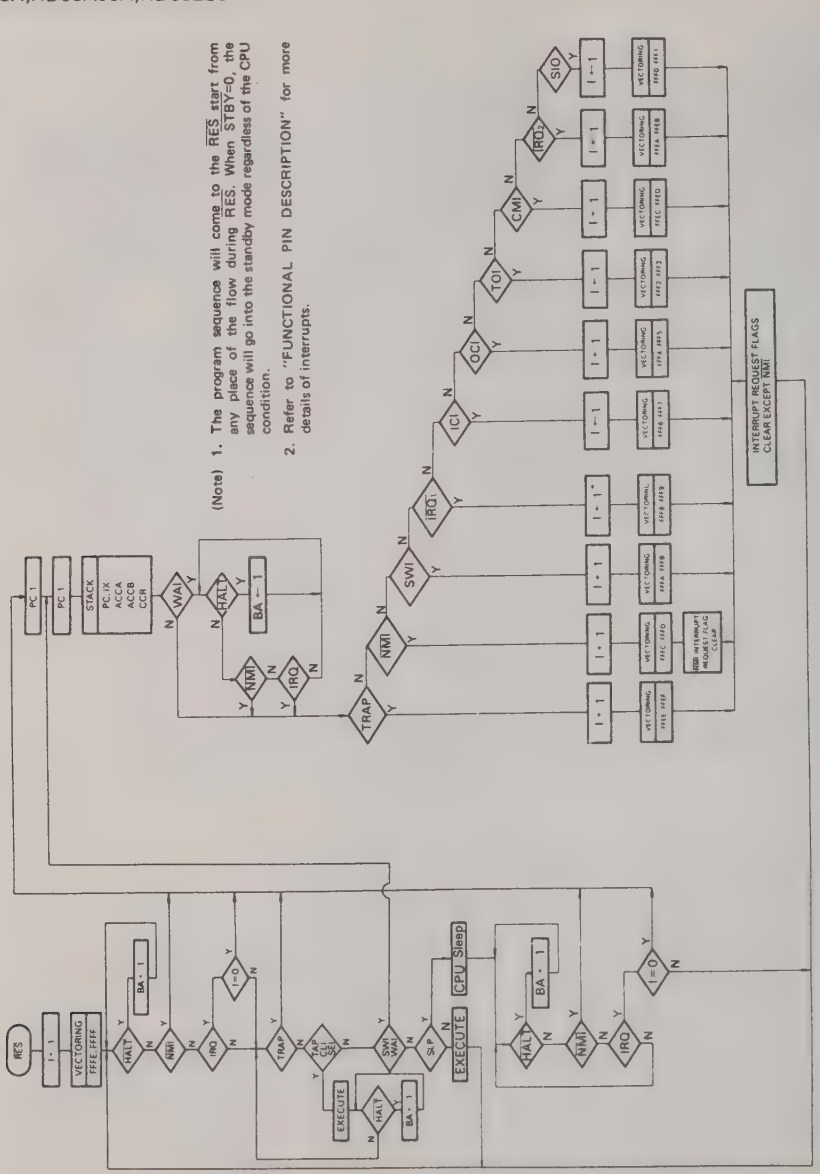


Figure 24 CPU Operation Mode Transition



(Note) 1. The program sequence will come to the RES start from any place of the flow during RES. When STBY=0, the sequence will go into the standby mode regardless of the CPU condition.

2. Refer to "FUNCTIONAL PIN DESCRIPTION" for more details of interrupts.

Figure 25 HD6303X System Flow Chart

Table 16 Cycle-by-Cycle Operation

Address Mode & Instructions		Cycles	Cycle #	Address Bus	R/W	$\overline{RD}$	$\overline{WR}$	$\overline{LIR}$	Data Bus
IMMEDIATE									
ADC	ADD	2	1	Op Code Address+1	1	0	1	1	Operand Data
AND	BIT		2	Op Code Address+2	1	0	1	0	Next Op Code
CMP	EOR								
LDA	ORA								
SBC	SUB								
ADDD	CPX	3	1	Op Code Address+1	1	0	1	1	Operand Data (MSB)
LDD	LDS		2	Op Code Address+2	1	0	1	1	Operand Data (LSB)
LDX	SUBD		3	Op Code Address+3	1	0	1	0	Next Op Code
DIRECT									
ADC	ADD	3	1	Op Code Address+1	1	0	1	1	Address of Operand (LSB)
AND	BIT		2	Address of Operand	1	0	1	1	Operand Data
CMP	EOR		3	Op Code Address+2	1	0	1	0	Next Op Code
LDA	ORA								
SBC	SUB								
STA		3	1	Op Code Address+1	1	0	1	1	Destination Address
			2	Destination Address	0	1	0	1	Accumulator Data
			3	Op Code Address+2	1	0	1	0	Next Op Code
ADDD	CPX	4	1	Op Code Address+1	1	0	1	1	Address of Operand (LSB)
LDD	LDS		2	Address of Operand	1	0	1	1	Operand Data (MSB)
LDX	SUBD		3	Address of Operand+1	1	0	1	1	Operand Data (LSB)
			4	Op Code Address+2	1	0	1	0	Next Op Code
STD	STS	4	1	Op Code Address+1	1	0	1	1	Destination Address (LSB)
STX			2	Destination Address	0	1	0	1	Register Data (MSB)
			3	Destination Address+1	0	1	0	1	Register Data (LSB)
			4	Op Code Address+2	1	0	1	0	Next Op Code
JSR		5	1	Op Code Address+1	1	0	1	1	Jump Address (LSB)
			2	FFFF	1	1	1	1	Restart Address (LSB)
			3	Stack Pointer	0	1	0	1	Return Address (LSB)
			4	Stack Pointer−1	0	1	0	1	Return Address (MSB)
			5	Jump Address	1	0	1	0	First Subroutine Op Code
TIM		4	1	Op Code Address+1	1	0	1	1	Immediate Data
			2	Op Code Address+2	1	0	1	1	Address of Operand (LSB)
			3	Address of Operand	1	0	1	1	Operand Data
			4	Op Code Address+3	1	0	1	0	Next Op Code
AIM	EIM	6	1	Op Code Address+1	1	0	1	1	Immediate Data
OIM			2	Op Code Address+2	1	0	1	1	Address of Operand (LSB)
			3	Address of Operand	1	0	1	1	Operand Data
			4	FFFF	1	1	1	1	Restart Address (LSB)
			5	Address of Operand	0	1	0	1	New Operand Data
			6	Op Code Address+3	1	0	1	0	Next Op Code

(Continued)

Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/$\overline{W}$	$\overline{RD}$	$\overline{WR}$	$\overline{LIR}$	Data Bus
INDEXED								
JMP	3	1	Op Code Address+1	1	0	1	1	Offset
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	Jump Address	1	0	1	0	First Op Code of Jump Routine
ADC ADD AND BIT CMP EOR LDA ORA SBC SUB TST	4	1	Op Code Address+1	1	0	1	1	Offset
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	IX+Offset	1	0	1	1	Operand Data
		4	Op Code Address+2	1	0	1	0	Next Op Code
STA	4	1	Op Code Address+1	1	0	1	1	Offset
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	IX+Offset	0	1	0	1	Accumulator Data
		4	Op Code Address+2	1	0	1	0	Next Op Code
ADDD CPX LDD LDS LDX SUBD	5	1	Op Code Address+1	1	0	1	1	Offset
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	IX+Offset	1	0	1	1	Operand Data (MSB)
		4	IX+Offset+1	1	0	1	1	Operand Data (LSB)
		5	Op Code Address+2	1	0	1	0	Next Op Code
STD STS STX	5	1	Op Code Address+1	1	0	1	1	Offset
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	IX+Offset	0	1	0	1	Register Data (MSB)
		4	IX+Offset+1	0	1	0	1	Register Data (LSB)
		5	Op Code Address+2	1	0	1	0	Next Op Code
JSR	5	1	Op Code Address+1	1	0	1	1	Offset
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	Stack Pointer	0	1	0	1	Return Address (LSB)
		4	Stack Pointer−1	0	1	0	1	Return Address (MSB)
		5	IX+Offset	1	0	1	0	First Subroutine Op Code
ASL ASR COM DEC INC LSR NEG ROL ROR	6	1	Op Code Address+1	1	0	1	1	Offset
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	IX+Offset	1	0	1	1	Operand Data
		4	FFFF	1	1	1	1	Restart Address (LSB)
		5	IX+Offset	0	1	0	1	New Operand Data
		6	Op Code Address+2	1	0	1	0	Next Op Code
TIM	5	1	Op Code Address+1	1	0	1	1	Immediate Data
		2	Op Code Address+2	1	0	1	1	Offset
		3	FFFF	1	1	1	1	Restart Address (LSB)
		4	IX+Offset	1	0	1	1	Operand Data
		5	Op Code Address+3	1	0	1	0	Next Op Code
CLR	5	1	Op Code Address+1	1	0	1	1	Offset
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	IX+Offset	1	0	1	1	Operand Data
		4	IX+Offset	0	1	0	1	00
		5	Op Code Address+2	1	0	1	0	Next Op Code
AIM EIM OIM	7	1	Op Code Address+1	1	0	1	1	Immediate Data
		2	Op Code Address+2	1	0	1	1	Offset
		3	FFFF	1	1	1	1	Restart Address (LSB)
		4	IX+Offset	1	0	1	1	Operand Data
		5	FFFF	1	1	1	1	Restart Address (LSB)
		6	IX+Offset	0	1	0	1	New Operand Data
		7	Op Code Address+3	1	0	1	0	Next Op Code

(Continued)

Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/$\overline{W}$	$\overline{RD}$	$\overline{WR}$	$\overline{LIR}$	Data Bus
EXTEND								
JMP	3	1	Op Code Address+1	1	0	1	1	Jump Address (MSB)
		2	Op Code Address+2	1	0	1	1	Jump Address (LSB)
		3	Jump Address	1	0	1	0	Next Op Code
ADC ADD TST AND BIT CMP EOR LDA ORA SBC SUB	4	1	Op Code Address+1	1	0	1	1	Address of Operand (MSB)
		2	Op Code Address+2	1	0	1	1	Address of Operand (LSB)
		3	Address of Operand	1	0	1	1	Operand Data
		4	Op Code Address+3	1	0	1	0	Next Op Code
STA	4	1	Op Code Address+1	1	0	1	1	Destination Address (MSB)
		2	Op Code Address+2	1	0	1	1	Destination Address (LSB)
		3	Destination Address	0	1	0	1	Accumulator Data
		4	Op Code Address+3	1	0	1	0	Next Op Code
ADDD CPX LDD LDS LDX SUBD	5	1	Op Code Address+1	1	0	1	1	Address of Operand (MSB)
		2	Op Code Address+2	1	0	1	1	Address of Operand (LSB)
		3	Address of Operand	1	0	1	1	Operand Data (MSB)
		4	Address of Operand+1	1	0	1	1	Operand Data (LSB)
		5	Op Code Address+3	1	0	1	0	Next Op Code
STD STS STX	5	1	Op Code Address+1	1	0	1	1	Destination Address (MSB)
		2	Op Code Address+2	1	0	1	1	Destination Address (LSB)
		3	Destination Address	0	1	0	1	Register Data (MSB)
		4	Destination Address+1	0	1	0	1	Register Data (LSB)
		5	Op Code Address+3	1	0	1	0	Next Op Code
JSR	6	1	Op Code Address+1	1	0	1	1	Jump Address (MSB)
		2	Op Code Address+2	1	0	1	1	Jump Address (LSB)
		3	FFFF	1	1	1	1	Restart Address (LSB)
		4	Stack Pointer	0	1	0	1	Return Address (LSB)
		5	Stack Pointer−1	0	1	0	1	Return Address (MSB)
		6	Jump Address	1	0	1	0	First Subroutine Op Code
ASL ASR COM DEC INC LSR NEG ROL ROR	6	1	Op Code Address+1	1	0	1	1	Address of Operand (MSB)
		2	Op Code Address+2	1	0	1	1	Address of Operand (LSB)
		3	Address of Operand	1	0	1	1	Operand Data
		4	FFFF	1	1	1	1	Restart Address (LSB)
		5	Address of Operand	0	1	0	1	New Operand Data
		6	Op Code Address+3	1	0	1	0	Next Op Code
CLR	5	1	Op Code Address+1	1	0	1	1	Address of Operand (MSB)
		2	Op Code Address+2	1	0	1	1	Address of Operand (LSB)
		3	Address of Operand	1	0	1	1	Operand Data
		4	Address of Operand	0	1	0	1	00
		5	Op Code Address+3	1	0	1	0	Next Op Code

(Continued)

Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/$\overline{W}$	$\overline{RD}$	$\overline{WR}$	$\overline{LIR}$	Data Bus
IMPLIED								
ABA ABX ASL ASLD ASR CBA CLC CLI CLR CLV COM DEC DES DEX INC INS INX LSR LSRD ROL ROR NOP SBA SEC SEI SEV TAB TAP TBA TPA TST TSX TXS	1	1	Op Code Address+1	1	0	1	0	Next Op Code
DAA XGDX	2	1	Op Code Address+1	1	0	1	0	Next Op Code
		2	FFFF	1	1	1	1	Restart Address (LSB)
PULA PULB	3	1	Op Code Address+1	1	0	1	0	Next Op Code
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	Stack Pointer+1	1	0	1	1	Data from Stack
PSHA PSHB	4	1	Op Code Address+1	1	0	1	1	Next Op Code
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	Stack Pointer	0	1	0	1	Accumulator Data
		4	Op Code Address+1	1	0	1	0	Next Op Code
PULX	4	1	Op Code Address+1	1	0	1	0	Next Op Code
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	Stack Pointer+1	1	0	1	1	Data from Stack (MSB)
		4	Stack Pointer+2	1	0	1	1	Data from Stack (LSB)
PSHX	5	1	Op Code Address+1	1	0	1	1	Next Op Code
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	Stack Pointer	0	1	0	1	Index Register (LSB)
		4	Stack Pointer−1	0	1	0	1	Index Register (MSB)
		5	Op Code Address+1	1	0	1	0	Next Op Code
RTS	5	1	Op Code Address+1	1	0	1	1	Next Op Code
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	Stack Pointer+1	1	0	1	1	Return Address (MSB)
		4	Stack Pointer+2	1	0	1	1	Return Address (LSB)
		5	Return Address	1	0	1	0	First Op Code of Return Routine
MUL	7	1	Op Code Address+1	1	0	1	0	Next Op Code
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	FFFF	1	1	1	1	Restart Address (LSB)
		4	FFFF	1	1	1	1	Restart Address (LSB)
		5	FFFF	1	1	1	1	Restart Address (LSB)
		6	FFFF	1	1	1	1	Restart Address (LSB)
		7	FFFF	1	1	1	1	Restart Address (LSB)

(Continued)

Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/$\overline{W}$	$\overline{RD}$	$\overline{WR}$	$\overline{LIR}$	Data Bus
IMPLIED								
WAI		1	Op Code Address+1	1	0	1	1	Next Op Code
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	Stack Pointer	0	1	0	1	Return Address (LSB)
		4	Stack Pointer−1	0	1	0	1	Return Address (MSB)
	9	5	Stack Pointer−2	0	1	0	1	Index Register (LSB)
		6	Stack Pointer−3	0	1	0	1	Index Register (MSB)
		7	Stack Pointer−4	0	1	0	1	Accumulator A
		8	Stack Pointer−5	0	1	0	1	Accumulator B
		9	Stack Pointer−6	0	1	0	1	Conditional Code Register
RTI		1	Op Code Address+1	1	0	1	1	Next Op Code
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	Stack Pointer+1	1	0	1	1	Conditional Code Register
		4	Stack Pointer+2	1	0	1	1	Accumulator B
		5	Stack Pointer+3	1	0	1	1	Accumulator A
	10	6	Stack Pointer+4	1	0	1	1	Index Register (MSB)
		7	Stack Pointer+5	1	0	1	1	Index Register (LSB)
		8	Stack Pointer+6	1	0	1	1	Return Address (MSB)
		9	Stack Pointer+7	1	0	1	1	Return Address (LSB)
		10	Return Address	1	0	1	0	First Op Code of Return Routine
SWI		1	Op Code Address+1	1	0	1	1	Next Op Code
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	Stack Pointer	0	1	0	1	Return Address (LSB)
		4	Stack Pointer−1	0	1	0	1	Return Address (MSB)
		5	Stack Pointer−2	0	1	0	1	Index Register (LSB)
		6	Stack Pointer−3	0	1	0	1	Index Register (MSB)
	12	7	Stack Pointer−4	0	1	0	1	Accumulator A
		8	Stack Pointer−5	0	1	0	1	Accumulator B
		9	Stack Pointer−6	0	1	0	1	Conditional Code Register
		10	Vector Address FFFA	1	0	1	1	Address of SWI Routine (MSB)
		11	Vector Address FFFB	1	0	1	1	Address of SWI Routine (LSB)
		12	Address of SWI Routine	1	0	1	0	First Op Code of SWI Routine
SLP		1	Op Code Address+1	1	0	1	1	Next Op Code
		2	FFFF	1	1	1	1	Restart Address (LSB)
		↑		|	|	|	|	
	4	Sleep		|	|	|	|	
		↓		↓	↓	↓	↓	↓
		3	FFFF	1	1	1	1	Restart Address (LSB)
		4	Op Code Address+1	1	0	1	0	Next Op Code
RELATIVE								
BCC BCS BEQ BGE BGT BHI BLE BLS BLT BMT BNE BPL BRA BRN BVC BVS	3	1	Op Code Address+1	1	0	1	1	Branch Offset
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	{ Branch Address ··· Test="1" { Op Code Address+1 ··· Test="0"	1	0	1	0	First Op Code of Branch Routine Next Op Code
BSR		1	Op Code Address+1	1	0	1	1	Offset
		2	FFFF	1	1	1	1	Restart Address (LSB)
	5	3	Stack Pointer	0	1	0	1	Return Address (LSB)
		4	Stack Pointer−1	0	1	0	1	Return Address (MSB)
		5	Branch Address	1	0	1	0	First Op Code of Subroutine

■ PRECAUTION TO THE BOARD DESIGN OF OSCILLATION CIRCUIT

As shown in Fig. 26, there is a case that the cross talk disturbs the normal oscillation if signal lines are put near the oscillation circuit. When designing a board, pay attention to this. Crystal and C_L must be put as near the HD6303X as possible.

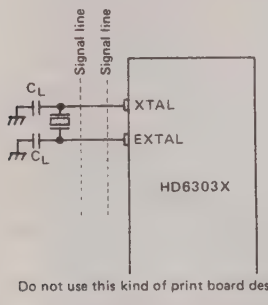


Figure 26 Precaution to the boad design of oscillation circuit

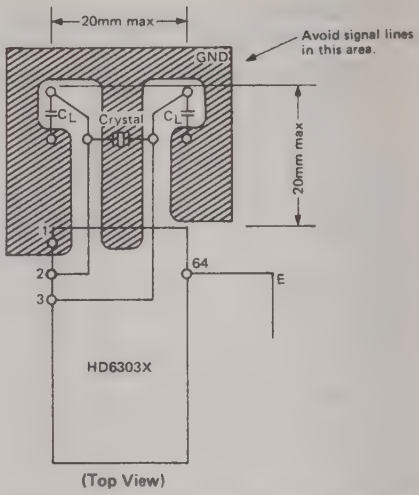


Figure 27 Example of Oscillation Circuits in Board Design

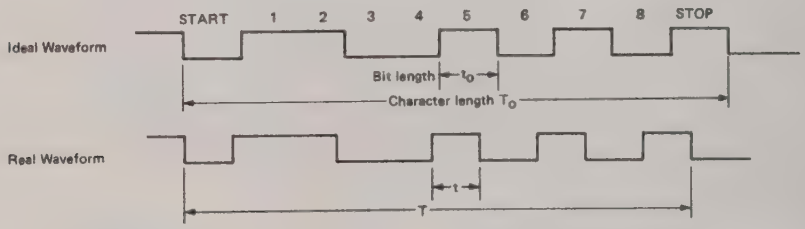
■ RECEIVE MARGIN OF THE SCI

Receive margin of the SCI contained in the HD6303X is shown in Table 17.

Note: SCI = Serial Communication Interface

Table 17

	Bit distortion tolerance (t−to) /to	Character distortion tolerance (T−To) /To
HD6303X	±43.7%	±4.37%



HD6303Y, HD63A03Y, HD63B03Y

CMOS MPU (Micro Processing Unit)

The HD6303Y is a CMOS 8-bit single-chip microprocessing unit which contains a CPU compatible with the CMOS 8-bit microcomputer HD6301V, 256 bytes of RAM, 24 parallel I/O pins, Serial Communication Interface (SCI) and two timers.

HD6303YP, HD63A03YP, HD63B03YP

(DP-64S)

■ FEATURES

- Instruction Set Compatible with the HD6301V1
- 256 Bytes of RAM
- 24 Parallel I/O Pins
- Parallel Handshake Interface (Port 6)
- Darlington Transistor Drive (Port 2, 6)
- 16-Bit Programmable Timer
  - Input Capture Register × 1
  - Free Running Counter × 1
  - Output Compare Register × 2
- 8-Bit Reloadable Timer
  - External Event Counter
  - Square Wave Generation
- Serial Communication Interface (SCI)
  - Asynchronous Mode (8 Transmit Formats, Hardware Parity)
  - Clocked Synchronous Mode
- Memory Ready
  - 3 Kinds of Memory Ready
- Halt
- Error Detection
  - (Address Error, Op-code Error)
- Interrupt — External 3, Internal 7
- Maximum 65k Bytes Address Space
- Low Power Dissipation Mode
  - Sleep Mode
  - Standby Mode (Hardware Standby, Software Standby)
- Minimum Instruction Execution Time — 0.5μs (f = 2MHz)
- Wide Range of Operation

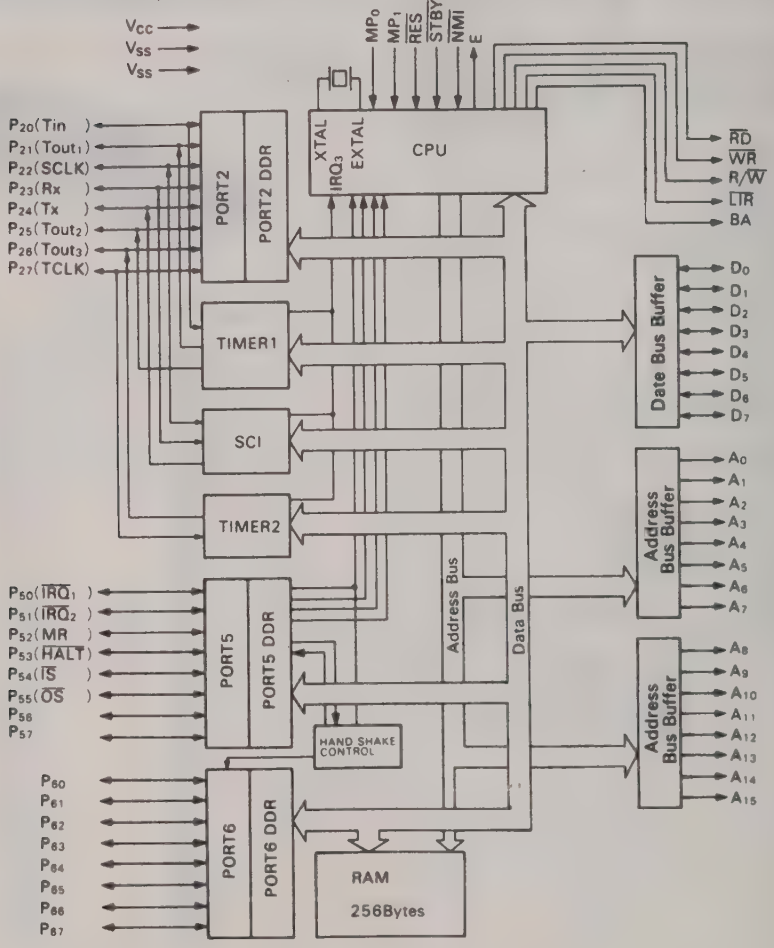
V_{CC}=3 to 5.5V (f=0.1 to 0.5MHz)

V_{CC}=5V±10% { f=0.1 to 1.0MHz : HD6303Y; f=0.1 to 1.5MHz : HD63A03Y; f=0.1 to 2.0MHz : HD63B03Y }

■ PIN ARRANGEMENT

(Top View)

■ **BLOCK DIAGRAM**



■ ABSOLUTE MAXIMUM RATINGS

Item	Symbol	Value	Unit
Supply Voltage	V_{CC}	$-0.3 \sim +7.0$	V
Input Voltage	V_{in}	$-0.3 \sim V_{CC}+0.3$	V
Operating Temperature	T_{opr}	$0 \sim +70$	°C
Storage Temperature	T_{stg}	$-55 \sim +150$	°C

(NOTE) This product has protection circuits in input terminal from high static electricity voltage and high electric field.
But be careful not to apply overvoltage more than maximum ratings to these high input impedance protection circuits. To assure the normal operation, we recommend V_{in}, V_{out}: $V_{SS} \leqq (V_{in}$ or $V_{out}) \leqq V_{CC}$.

■ ELECTRICAL CHARACTERISTICS

• DC CHARACTERISTICS ($V_{CC} = 5.0V \pm 10\%$, $V_{SS} = 0V$, $Ta = 0 \sim +70°C$, unless otherwise noted.)

Item		Symbol	Test Condition	min	typ	max	Unit
Input "High" Voltage	$\overline{RES}$, $\overline{STBY}$	V_{IH}		$V_{CC}-0.5$	–	$V_{CC}+0.3$	V
	EXTAL			$V_{CC} \times 0.7$	–		
	Other Inputs			2.0	–		
Input "Low" Voltage	All Inputs	V_{IL}		-0.3	–	0.8	V
Input Leakage Current	$\overline{NMI}$, $\overline{RES}$, $\overline{STBY}$, MP_0, MP_1	$\lvert I_{in} \rvert$	$V_{in} = 0.5 \sim V_{CC}-0.5V$	–	–	1.0	μA
Three State Leakage Current	$A_0 \sim A_{15}$, $D_0 \sim D_7$, $\overline{RD}$, $\overline{WR}$, R/$\overline{W}$, Ports 2, 5, 6	$\lvert I_{TSI} \rvert$	$V_{in} = 0.5 \sim V_{CC}-0.5V$	–	–	1.0	μA
Output "High" Voltage	All Outputs	V_{OH}	$I_{OH} = -200\mu A$	2.4	–	–	V
			$I_{OH} = -10\mu A$	$V_{CC}-0.7$	–	–	V
Output "Low" Voltage	All Outputs	V_{OL}	$I_{OL} = 1.6mA$	–	–	0.4	V
Darlington Drive Current	Ports 2, 6	$-I_{OH}$	$V_{out} = 1.5V$	1.0	–	10.0	mA
Input Capacitance	All Inputs	C_{in}	$V_{in} = 0V$, $f = 1MHz$, $Ta = 25°C$	–	–	12.5	pF
Standby Current	Non Operation	I_{STB}		–	3.0	15.0	μA
Current Dissipation*		I_{SLP}	Sleeping (f = 1MHz**)	–	1.5	3.0	mA
			Sleeping (f = 1.5MHz**)	–	2.3	4.5	mA
			Sleeping (f = 2MHz**)	–	3.0	6.0	mA
		I_{CC}	Operating (f = 1MHz**)	–	7.0	10.0	mA
			Operating (f = 1.5MHz**)	–	10.5	15.0	mA
			Operating (f = 2MHz**)	–	14.0	20.0	mA
RAM Standby Voltage		V_{RAM}		2.0	–	–	V

* V_{IH} min = V_{CC} – 1.0V, V_{IL} max = 0.8V (All output terminals are at no load.)

** Current Dissipation of the operating or sleeping condition is proportional to the operating frequency. So the typ. or max. values about Current Dissipations at X MHz operation are decided according to the following formula:

typ. value (f = X MHz) = typ. value (f = 1MHz) × X
max. value (f = X MHz) = max. value (f = 1MHz) × X
(both the sleeping and operating)

● **AC CHARACTERISTICS** (V_{CC} = 5.0V±10%, V_{SS} = 0V, T_a = 0~+70°C, unless otherwise noted.)

BUS TIMING

Item		Symbol	Test Condition	HD6303Y			HD63A03Y			HD63B03Y			Unit
				min	typ	max	min	typ	max	min	typ	max	
Cycle Time		t_{cyc}	Fig. 1	1	–	10	0.666	–	10	0.5	–	10	μs
Enable Rise Time		t_{Er}		–	–	25	–	–	25	–	–	25	ns
Enable Fall Time		t_{Ef}		–	–	25	–	–	25	–	–	25	ns
Enable Pulse Width "High" Level*		PW_{EH}		450	–	–	300	–	–	220	–	–	ns
Enable Pulse Width "Low" Level*		PW_{EL}		450	–	–	300	–	–	220	–	–	ns
Address, R/$\overline{W}$ Delay Time*		t_{AD}		–	–	250	–	–	190	–	–	160	ns
Data Delay Time	Write	t_{DDW}		–	–	200	–	–	160	–	–	120	ns
Data Set-up Time	Read	t_{DSR}		80	–	–	70	–	–	60	–	–	ns
Address, R/$\overline{W}$ Hold Time*		t_{AH}		80	–	–	50	–	–	40	–	–	ns
Data Hold Time	Write*	t_{HW}		70	–	–	50	–	–	40	–	–	ns
	Read	t_{HR}		0	–	–	0	–	–	0	–	–	ns
$\overline{RD}$, $\overline{WR}$ Pulse Width*		PW_{RW}		450	–	–	300	–	–	220	–	–	ns
$\overline{RD}$, $\overline{WR}$ Delay Time		t_{RWD}		–	–	40	–	–	40	–	–	40	ns
$\overline{RD}$, $\overline{WR}$ Hold Time		t_{HRW}		–	–	20	–	–	20	–	–	20	ns
$\overline{LIR}$ Delay Time		t_{DLR}		–	–	200	–	–	160	–	–	120	ns
$\overline{LIR}$ Hold Time		t_{HLR}		10	–	–	10	–	–	10	–	–	ns
MR Set-up Time*		t_{SMR}	Fig. 2	400	–	–	280	–	–	230	–	–	ns
MR Hold Time*		t_{HMR}		–	–	100	–	–	70	–	–	50	ns
E Clock Pulse Width at MR		PW_{EMR}		–	–	9	–	–	9	–	–	9	μs
Processor Control Set-up Time		t_{PCS}	Fig. 3, 13, 14	200	–	–	200	–	–	200	–	–	ns
Processor Control Rise Time		t_{PCr}	Fig. 2, 3	–	–	100	–	–	100	–	–	100	ns
Processor Control Fall Time		t_{PCf}		–	–	100	–	–	100	–	–	100	ns
BA Delay Time		t_{BA}	Fig. 3	–	–	250	–	–	190	–	–	160	ns
Oscillator Stabilization Time		t_{RC}	Fig. 14	20	–	–	20	–	–	20	–	–	ms
Reset Pulse Width		PW_{RST}		3	–	–	3	–	–	3	–	–	t_{cyc}

* These timings change in approximate proportion to t_{cyc}. The figures in this characteristics represent those when t_{cyc} is minimum (= in the highest speed operation).

Peripheral Port Timing

Item		Symbol	Test Condition	HD6303Y			HD63A03Y			HD63B03Y			Unit
				min	typ	max	min	typ	max	min	typ	max	
Peripheral Data Set Up Time	Port 2, 5, 6	t_{PDSU}	Fig. 5	200	–	–	200	–	–	200	–	–	ns
Peripheral Data Hold Time	Port 2, 5, 6	t_{PDH}		200	–	–	200	–	–	200	–	–	ns
Delay Time (From Enable Fall Edge to Peripheral Output)	Port 2, 5, 6	t_{PWD}	Fig. 6	–	–	300	–	–	300	–	–	300	ns
Input Strobe Pulse Width		t_{PWIS}	Fig. 10	200	–	–	200	–	–	200	–	–	ns
Input Data Hold Time	Port 6	t_{IH}		150	–	–	150	–	–	150	–	–	ns
Input Data Set-Up Time	Port 6	t_{IS}		100	–	–	100	–	–	100	–	–	ns
Output Strobe Delay Time		t_{OSD1} / t_{OSD2}	Fig. 11	–	–	200	–	–	200	–	–	200	ns

TIMER, SCI TIMING

Item		Symbol	Test Condition	HD6303Y			HD63A03Y			HD63B03Y			Unit
				min	typ	max	min	typ	max	min	typ	max	
Timer 1 Input Pulse Width		t_{PWT}	Fig. 9	2.0	–	–	2.0	–	–	2.0	–	–	t_{cyc}
Delay Time (Enable Positive Transition to Timer Output)		t_{TOD}	Fig. 7, 8	–	–	400	–	–	400	–	–	400	ns
SCI Input Clock Cycle	Async. Mode	t_{Scyc}	Fig. 9	1.0	–	–	1.0	–	–	1.0	–	–	t_{cyc}
	Clock Sync.		Fig. 4	2.0	–	–	2.0	–	–	2.0	–	–	t_{cyc}
SCI Transmit Data Delay Time (Clock Sync. Mode)		t_{TXD}	Fig. 4	–	–	220	–	–	220	–	–	220	ns
SCI Receive Data Set-up Time (Clock Sync. Mode)		t_{SRX}		260	–	–	260	–	–	260	–	–	ns
SCI Receive Data Hold Time (Clock Sync. Mode)		t_{HRX}		100	–	–	100	–	–	100	–	–	ns
SCI Input Clock Pulse Width		t_{PWSCK}	Fig. 9	0.4	–	0.6	0.4	–	0.6	0.4	–	0.6	t_{Scyc}
Timer 2 Input Clock Cycle		t_{tcyc}		2.0	–	–	2.0	–	–	2.0	–	–	t_{cyc}
Timer 2 Input Clock Pulse Width		t_{PWTCK}		200	–	–	200	–	–	200	–	–	ns
Timer 1·2, SCI Input Clock Rise Time		t_{CKr}		–	–	100	–	–	100	–	–	100	ns
Timer 1·2, SCI Input Clock Fall Time		t_{CKf}		–	–	100	–	–	100	–	–	100	ns

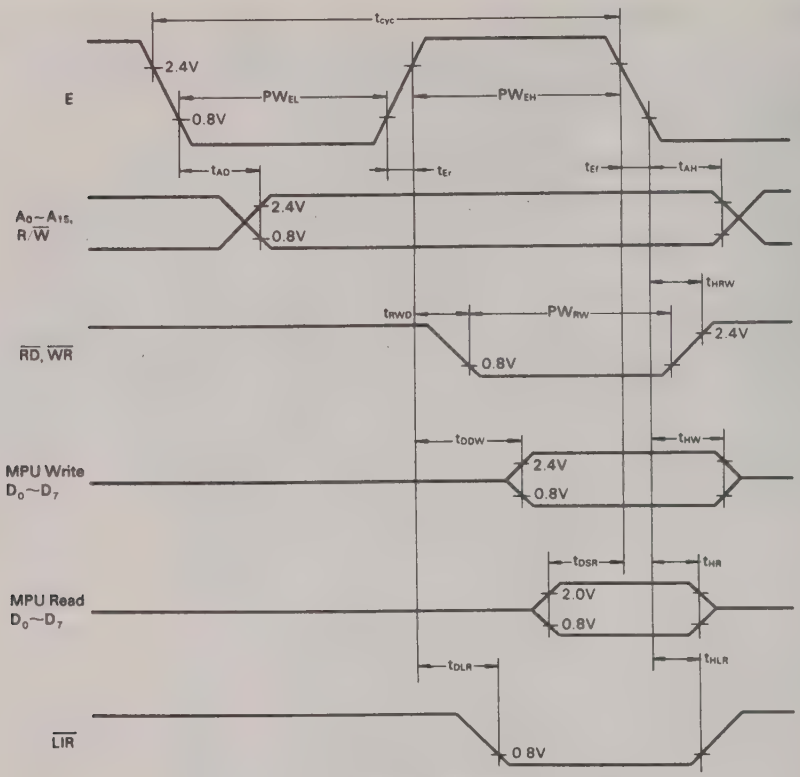


Figure 1 Bus Timing

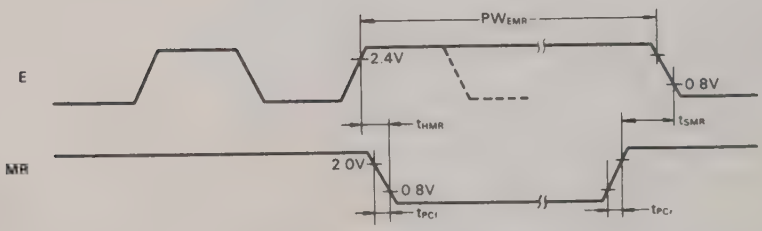


Figure 2 Memory Ready and E Clock Timing

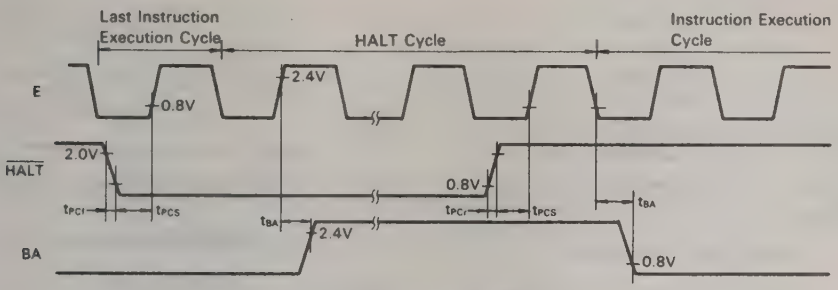
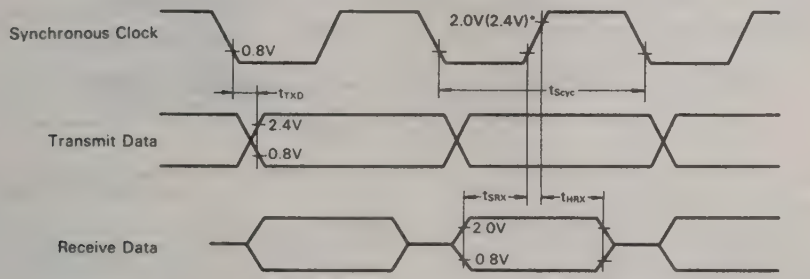


Figure 3 $\overline{\text{HALT}}$ and BA Timing



*2.0V is high level when clock input.
2.4V is high level when clock output.

Figure 4 SCI Clocked Synchronous Timing

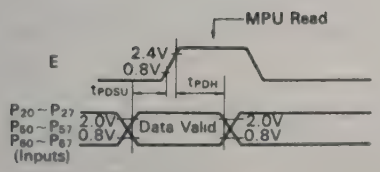


Figure 5 Port Data Set-up and Hold Times (MPU Read)

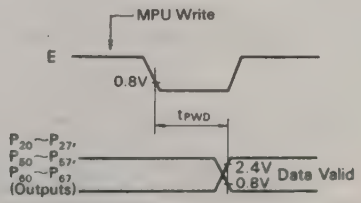


Figure 6 Port Data Delay Times (MPU Write)

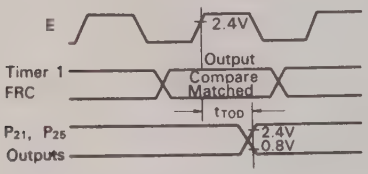
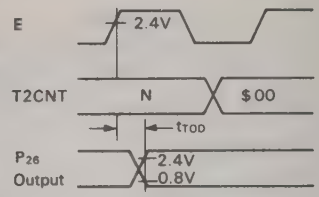
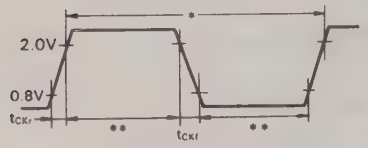


Figure 7 Timer 1 Output Timing



(TCONR=N)

Figure 8 Timer 2 Output Timing



*Timer 2 ; ttcyc
SCI ; tScyc

**Timer 1 ; tPWT
Timer 2 ; tPWTCK
SCI ; tPWSCK

Figure 9 Timer 1·2, SCI Input Clock Timing

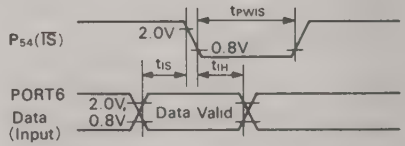


Figure 10 Port 6 Input Latch Timing

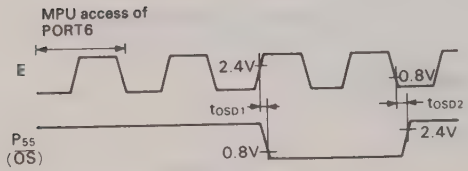
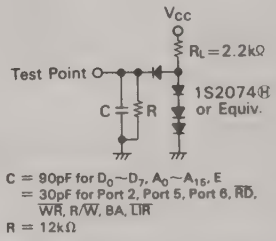


Figure 11 Output Strobe Timing



C = 90pF for D_0~D_7, A_0~A_{15}, E
= 30pF for Port 2, Port 5, Port 6, $\overline{RD}$, $\overline{WR}$, R/$\overline{W}$, BA, $\overline{LIR}$
R = 12kΩ

Figure 12 Bus Timing Test Loads (TTL Load)

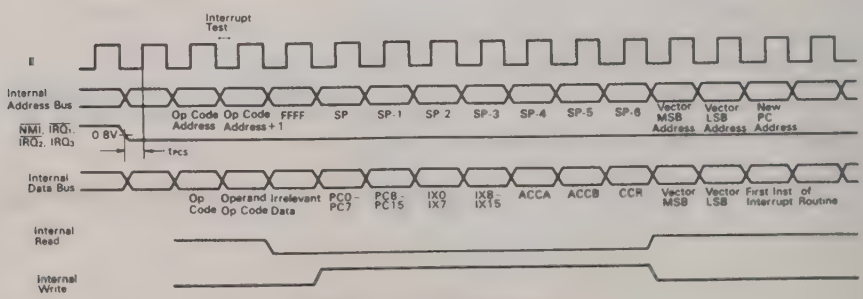


Figure 13 Interrupt Sequence

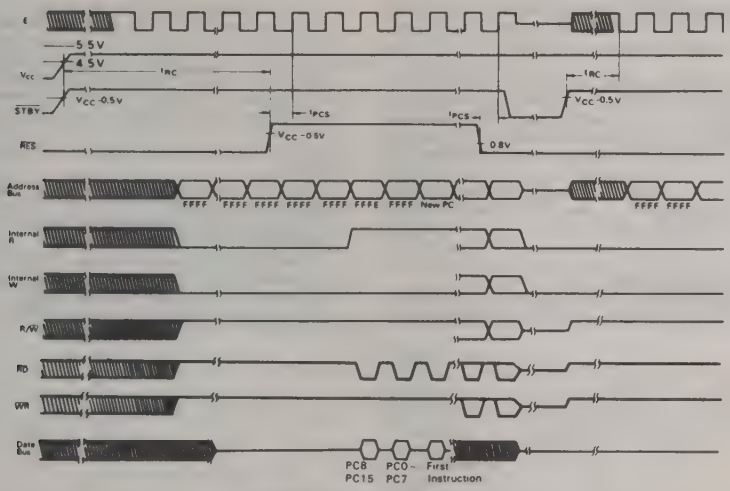


Figure 14 Reset Timing

■ FUNCTIONAL PIN DESCRIPTION

• V_{CC}, V_{SS}

V_{CC} and V_{SS} provide power to the MPU with 5V±10% supply. In the case of low speed operation (fmax=500kHz), the MPU can operate with 3 to 5.5 volts. Two V_{SS} pins should be tied to ground.

• XTAL, EXTAL

These two pins interface with an AT-cut parallel resonant crystal. Divide-by-four circuit is on chip, so if 4MHz crystal oscillator is used, the system clock is 1MHz for example.

EXTAL pin can be drived by the external clock with 45% to 55% duty. The system clock which is one fourth frequency of the external clock is generated in the LSI. The external clock frequency should be less than four times of the maximum operating frequency. When using the external clock, XTAL pin should be open. Fig. 15 shows examples of connection circuit. The crystal and C_{L1}, C_{L2} should be mounted as close as possible to XTAL and EXTAL pins. Any line must not cross the line between the crystal oscillator and XTAL, EXTAL.

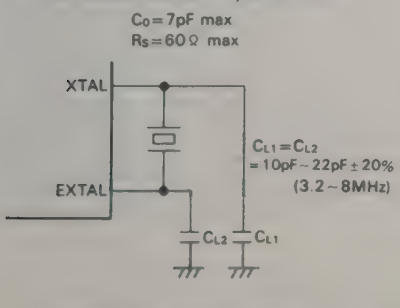


Figure 15 Connection Circuit

• $\overline{STBY}$

This pin makes the MPU standby mode. In "Low" level, the oscillation stops and the internal clock is stabilized to make reset condition. To retain the contents of RAM at standby mode, "0" should be written into RAM enable bit (RAME). RAME is the bit 6 of the RAM/port 5 control register at $0014. RAM is disabled by this operation and its contents is sustained.

Refer to "LOW POWER DISSIPATION MODE" for the standby mode.

• Reset ($\overline{RES}$)

This pin resets the MPU from power OFF state and provides a startup procedure. During power-on, $\overline{RES}$ pin must be held "Low" level for at least 20ms.

The CPU registers (accumulator, index register, stack pointer, condition code register except for interrupt mask bit), RAM and the data register of ports are not initialized during reset, so their contents are undefined in this procedure.

To reset the MPU during operation, $\overline{RES}$ should be held "Low" for at least 3 system-clock cycles. At the 3rd cycle during "Low" level, all the address buses become "High". When $\overline{RES}$ remains "Low", the address buses keep "High". If $\overline{RES}$ becomes "High", the MPU starts the next operation.

(1) Latch the value of the mode program pins; MP_0 and MP_1.
(2) Initialize each internal register (Refer to Table 4).
(3) Set the interrupt mask bit. For the CPU to recognize the maskable interrupts $\overline{IRQ_1}$, $\overline{IRQ_2}$ and IRQ_3, this bit should be cleared in advance.
(4) Put the contents (=start address) of the last two addresses ($FFFE, $FFFF) into the program counter and start the program from this address. (Refer to Table 1).

• Enable (E)

This pin provides a TTL-compatible system clock to external circuits. Its frequency is one fourth that of the crystal oscillator or external clock. This pin can drive one TTL load and 90pF capacitance.

• Non-Maskable Interrupt ($\overline{NMI}$)

When the falling edge of the input signal is detected at this pin, the CPU begins non-maskable interrupt sequence internally. As

well as the $\overline{IRQ}$ mentioned below, the instruction being executed at $\overline{NMI}$ signal detection will proceed to its compeletion. The interrupt mask bit of the condition code register doesn't affect non-maskable interrupt at all.

In response to an $\overline{NMI}$ interrupt, the contents of the program counter, index register, accumulators and condition code register will be saved onto the stack. Upon completion of this sequence, a vector is fetched from $FFFC and $FFFD to transfer their contents into the program counter and branch to the non-maskable interrupt service routine.

(Note) At reset start, the stack pointer should be initialized on an appropriate memory area and then the falling edge be input to $\overline{NMI}$ pin.

- **Interrupt Request ($\overline{IRQ_1}$, $\overline{IRQ_2}$)**

These are level-sensitive pins which request an internal interrupt sequence to the CPU. At interrupt request, the CPU will complete the current instruction before the acceptance of the request. Unless the interrupt mask in the condition code register is set, the CPU starts an interrupt sequence; if set, the interrupt request will be ignored. When the sequence starts, the contents of the program counter, index register, accumulators and condition code register will be saved onto the stack, then the CPU sets the interrupt mask bit and will not acknowledge the maskable request. During the last cycle, the CPU fetches vectors depicted in Table 1 and transfers their contents to the program counter and branches to the service routine.

The CPU uses the external interrupt pins ($\overline{IRQ_1}$ and $\overline{IRQ_2}$) also as port pins P_{50} and P_{51}, so it provides an enable bit to Bit 0 and 1 of the RAM port 5 control register at $0014. Refer to "RAM/PORT 5 CONTROL REGISTER" for the details.

When one of the internal interrupts, ICI, OCI, TOI, CMI or SIO is generated, the CPU produces internal interrupt signal (IRQ_3). IRQ_3 functions just the same as $\overline{IRQ_1}$ or $\overline{IRQ_2}$ except for its vector address. Fig. 16 shows the block diagram of the interrupt circuit.

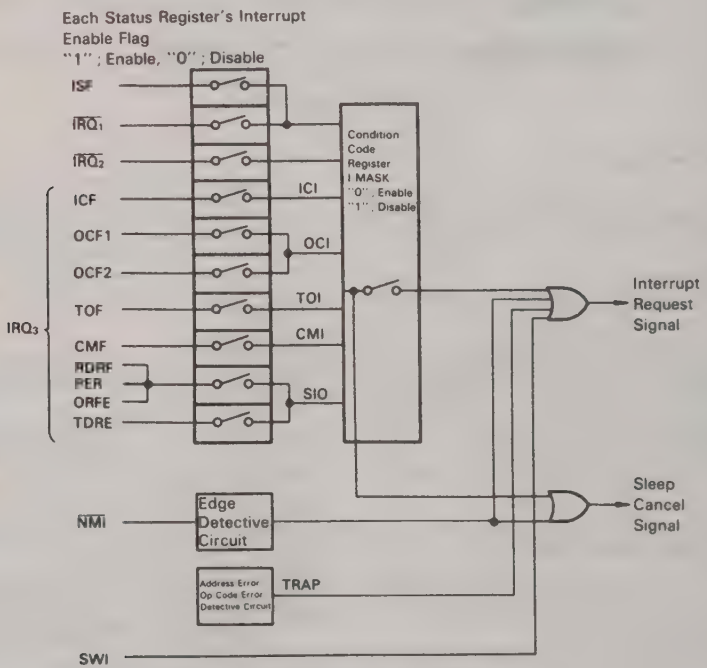


Figure 16 Interrupt Circuit Block Diagram

Table 1 Interrupt Vector Memory Map

Priority	Vector MSB	Vector LSB	Interrupt
Highest	FFFE	FFFF	$\overline{RES}$
	FFEE	FFEF	TRAP
	FFFC	FFFD	$\overline{NMI}$
	FFFA	FFFB	SWI (Software Interrupt)
	FFF8	FFF9	$\overline{IRQ_1}$, ISF (port 6 Input Strobe)
	FFF6	FFF7	ICI (Timer 1 Input Capture)
	FFF4	FFF5	OCI (Timer 1 Output Compare 1, 2)
	FFF2	FFF3	TOI (Timer 1 Overflow)
	FFEC	FFED	CMI (Timer 2 Counter Match)
	FFEA	FFEB	$\overline{IRQ_2}$
Lowest	FFF0	FFF1	SIO (RDRF+ORFE+TDRE+PER)

- **Mode Program (MP_0, MP_1)**
  Set MP_0 "High" and MP_1 "Low".

- **Read/Write (R/$\overline{W}$)**
  This signal, usually be in read state ("High"), shows whether the CPU is in read ("High") or write ("Low") state to the peripheral or memory devices. This can drive one TTL load and 30pF capacitance.

- **$\overline{RD}$, $\overline{WR}$**
  These signals show active low outputs when the CPU is reading/writing to the peripherals or memories. This enables the CPU easy to access the peripheral LSI with $\overline{RD}$ and $\overline{WR}$ input pins. These pins can drive one TTL load and 30pF capacitance.

- **Load Instruction Register ($\overline{LIR}$)**
  This signal shows the instruction opecode being on data bus (active low). This pin can drive one TTL load and 30pF capacitance.

- **Memory Ready (MR; P_{62})**
  This is the input control signal which stretches the system clock's "High" period to access low-speed memories. HD6303Y can select three kinds of low-speed memory access method by RAM/Port 5 Control Register's MRE bit and AMRE bit. In the case that CPU accesses low-speed memories by the external MR signal (MRE="1", AMRE="0"), the system clock operates in normal sequence when this signal is in "High".

  But this signal in "Low", the "High" period of the system clock will be stretched depending on its "Low" level duration in integral multiples of the cycle time. This allows the CPU to interface with low-speed memories (See Fig. 2). Up to 9μs can be stretched.

  During internal address space access or nonvalid memory access, MR is prohibited internally to prevent decrease of operation speed. Even in the halt state, MR can also stretch "High" period of system clock to allow peripheral devices to access low-speed memories. Refer to "RAM/PORT 5 CONTROL REGISTER" for more details.

- **Halt ($\overline{HALT}$; P_{63})**
  This is an input control signal to stop instruction execution and to release buses. When this signal switches to "Low", the CPU stops to enter into the halt state after having executed the present instruction. When entering into the halt state, it makes BA "High" and also an address bus, data bus, $\overline{RD}$, $\overline{WR}$, R/$\overline{W}$ high impedance. When an interrupt is generated in the halt state, the CPU uses the interrupt handler after the halt is cancelled. When halted during the sleep state, the CPU keeps the sleep state, while BA is "High" and releases the buses. Then the CPU returns to the previous sleep state when the $\overline{HALT}$ signal becomes "High".
  (Note) Please don't switch the $\overline{HALT}$ signal to "Low" when the CPU executes the WAI instruction and is in the interrupt wait state to avoid the trouble of the CPU's operation after the halt is cancelled.

- **Bus Available (BA)**
  This is an output control signal which is normally "Low" but "High" when the CPU accepts $\overline{HALT}$ and releases the buses. The HD6800 and HD6802 make BA "High" and release the buses at WAI execution, while the HD6303Y doesn't make BA "High" under the same condition.

■ **PORT**

The HD6303Y provides three 8-bit I/O ports. Each port provides Data Direction Register (DDR) which controls the I/O state by the bit.

Table 2 Port and Data Direction Register Address

Port	Port Address	Data Direction Register
Port 2	$0003	$0001
Port 5	$0015	$0020
Port 6	$0017	$0016

- **Port 2**
  An 8-bit I/O port. Port 2 DDR (P2DDR) controls the I/O state. This port provides DDR corresponding to each bit and can define input or output by the bit ("0" for input, "1" for output).

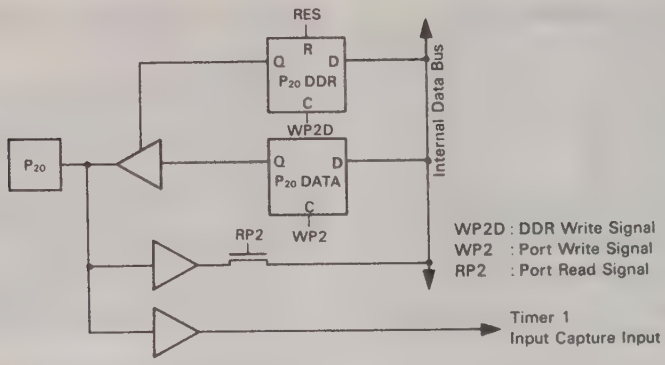
  As Port 2 DDR is cleared during reset, it will be an input port.

  Port 2 is also used as an I/O pin for timer 1, Timer 2 and the SCI. Pins for Timers and the SCI set or reset each DDR depending on their functions and become I/O pins. When port 2 functions as an I/O port after used as I/O pins of the timers or the SCI, the I/O direction of the pins remain as it is used as the I/O pin of timer and SCI.

  Port 2 can drive one TTL load and 30pF capacitance. This port can produce 1mA when V_{out}=1.5V to drive directly the base of Darlington transistor.

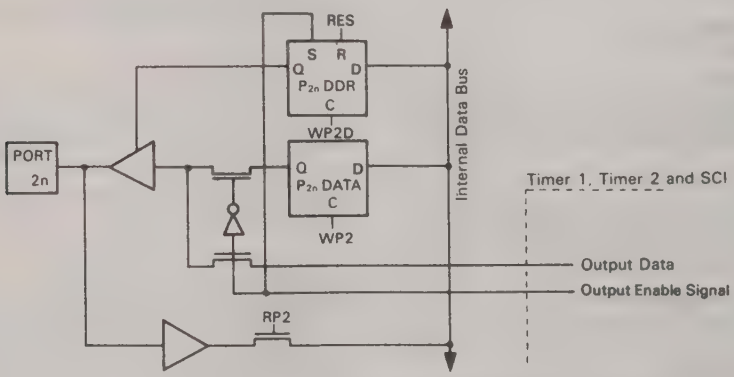
P_{20} (TIn)

P_{20} is also used as an external input pin for the input-capture. This pin is an I/O port which is an input or output as defined by the Data Direction Register (P_{20}DDR) ("0" for an input and "1" for an output). Then either a signal to or from P_{20} ("to" for an output port, "from" for an input port) is always input to the Timer 1 input capture.



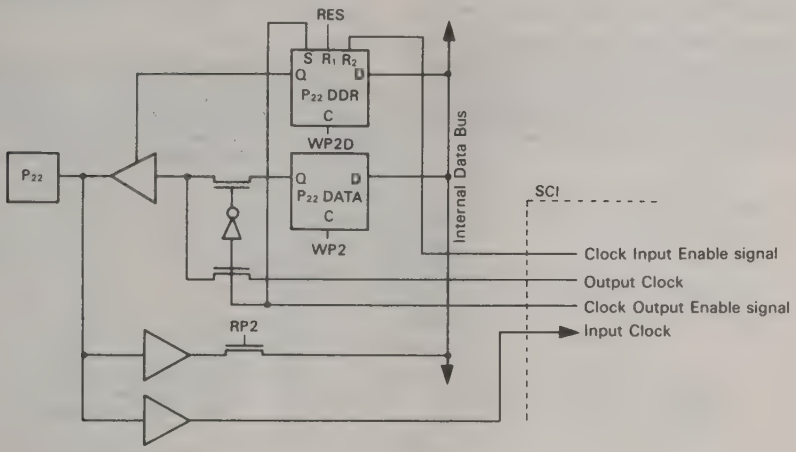
P_{21} (Tout 1), P_{24} (Tx), P_{25} (Tout 2), P_{26} (Tout 3)

These four pins can be also used as output pins for Timer 1, Timer 2 and a transmit output of the SCI. Timer 1, and the SCI have a register which enables output. By setting these registers, they automatically will be output pins of timer or the SCI.



P_{22} (SCLK)

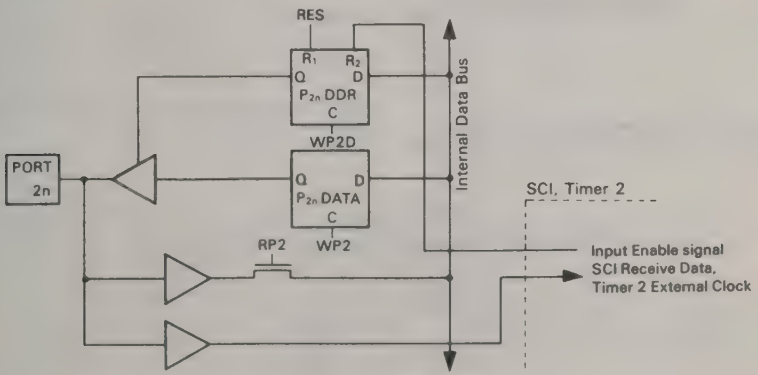
P_{22} is also used as a clock I/O pin for the SCI. It is selected as a clock input or output pin by the operating mode of the SCI. It is usable as an I/O port when the SCI has no clock input or output (as an output port if P_{22} DDR=1, as an input port if P_{22} DDR=0).

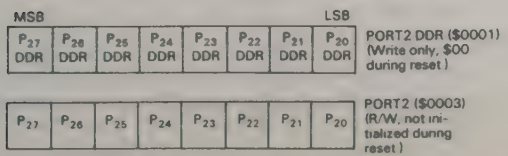


P_{23} (Rx), P_{27} (TCLK)

P_{23} and P_{27} are also used as received data input pins for the SCI and external clock input pins for Timer 2. The SCI and Timer 2 have registers which enable input. If the registers are set, the DDR (P_{23} DDR, P_{27} DDR) are cleared and P_{23} and P_{27} will be input pins for Rx and TCLK.

Since the SCI will be a clocked synchronous mode by an external clock-input during reset, the DDR of P_{22} is cleared automatically and P_{22} is an input port. Set the SCI to a mode where P_{22} is not used (CC0 or CC1 of the RMC Register is "0" or "1" respectively) and write "1" to the P_{22} DDR to make P_{22} an output port.





- **Port 5**

An 8-bit I/O port. The DDR of port 5 controls I/O state. Each bit of port 5 has a DDR which defines I/O state ("0" for input and "1" for output).

During reset, the DDR of port 5 is cleared and port 5 becomes an input port.

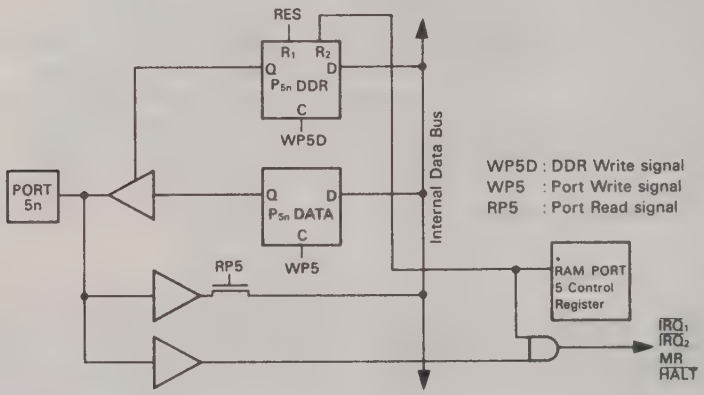
Port 5 is also usable as $\overline{IRQ_1}$, $\overline{IRQ_2}$, $\overline{HALT}$, MR and the strobed signal of port 6 for handshake ($\overline{IS}$, $\overline{OS}$). It is set to input or output automatically if it is used as these control signal pins (except P_{54}, $\overline{IS}$). Since the DDR of port 5, as is port 2, is set or reset by the control signal, I/O directions of the I/O ports are retained after the control signal is disabled. Port 5 can drive one TTL load and 90pF capacitance.

P_{50} ($\overline{IRQ_1}$), P_{51} ($\overline{IRQ_2}$)

P_{50} and P_{51} are also usable as interrupt pins. The RAM/port 5 control registers of $\overline{IRQ_1}$ and $\overline{IRQ_2}$ have enable bits (IQ1E, IQ2E). When these bits are set to "1", P_{50} and P_{51} will automatically be interrupt input pins.

P_{52} (MR), P_{53} ($\overline{HALT}$)

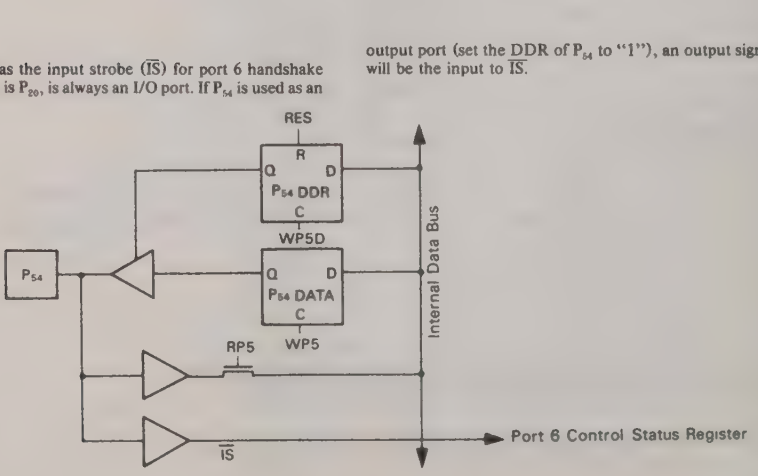
P_{52} and P_{53} are also usable as MR and $\overline{HALT}$ inputs. MR and $\overline{HALT}$ have enable bits (MRE, HLTE) in the RAM/Port 5 Control Register as $\overline{IRQ_1}$ and $\overline{IRQ_2}$. Since MRE is cleared during reset, P_{52} is usable as an I/O port, and HLTE is set during reset, the DDR of P_{53} will be automatically reset to be a $\overline{HALT}$ input pin. HLTE of the RAM/Port 5 Control Register has to be cleared to use P_{53} as an I/O port.



* Initializing value during reset;
1RQ1E = "0", 1RQ2E = "0", MRE = "0", HLTE = "1"

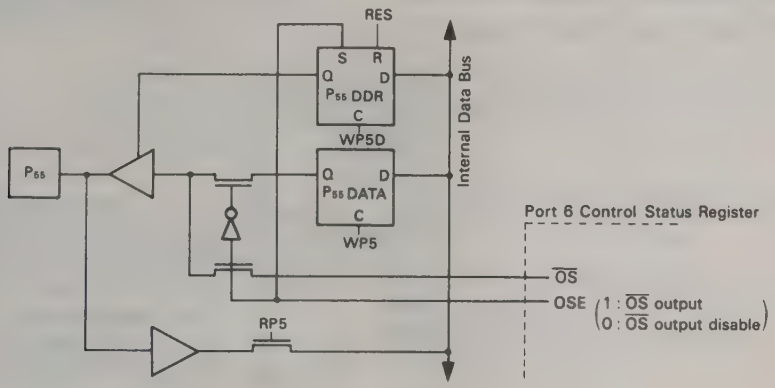
P_{54} ($\overline{IS}$)

P_{54} is also usable as the input strobe ($\overline{IS}$) for port 6 handshake interface. This pin, as is P_{20}, is always an I/O port. If P_{54} is used as an output port (set the DDR of P_{54} to "1"), an output signal from P_{64} will be the input to $\overline{IS}$.



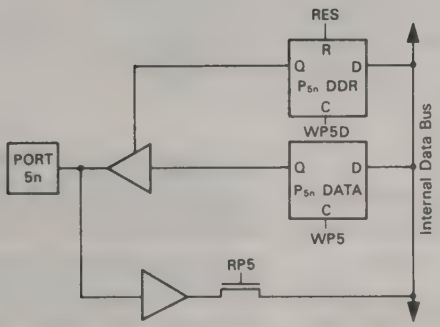
P_{55} ($\overline{OS}$)

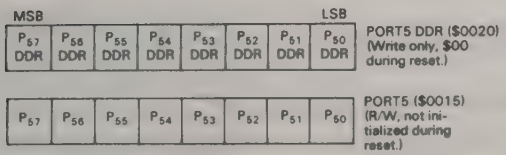
P_{55} is also usable as the output strobe ($\overline{OS}$) for port 6 handshake interface. It will be an I/O port during reset, and an $\overline{OS}$ output pin by setting the $\overline{OS}$ enable register (OSE) of the port 6 Control Status Register (P6CSR).



P_{56}, P_{57}

P_{56} and P_{57} are I/O ports.



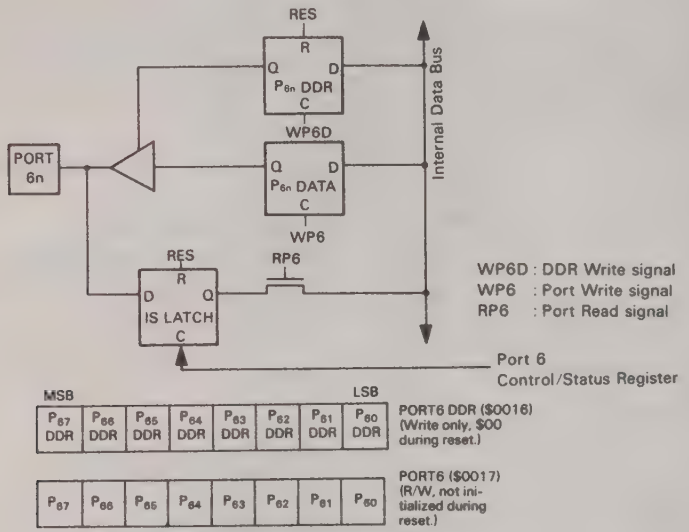


- **Port 6**

8-bit I/O port. Port 6 DDR controls I/O state. Each bit of port 6 has a DDR and designates input or output ("0" for input, "1" for output). During reset, Port 6 DDR is cleared and port 6 becomes an input port.

Port 6 controls parallel handshake interface besides functions as an I/O port. Therefore, it provides DDRs to control and IS LATCH to latch the input data.

Port 6 can drive one TTL load and 30pF capacitance. It can drive directly the base of Darlington transistor as port 2.



■ BUS

- **Address Bus ($A_0 \sim A_{15}$)**

Address Bus ($A_0 \sim A_{15}$) is used for addressing the memory and peripheral LSI.

This bus can interface with the bus of HMCS 6800 and drive one TTL load and 90pF capacitance.

- **Data Bus ($D_0 \sim D_7$)**

8-bit parallel data bus for data transmit between the memory or peripheral LSI. This bus can drive one TTL load and 90pF capacitance.

■ RAM/PORT 5 CONTROL REGISTER

The control register located at $0014 controls on-chip RAM and port 5.

RAM/Port 5 Control Register (RP5CR)

7	6	5	4	3	2	1	0	
STBY PWR	RAME	STBY FLAG	AMR E	HLTE	MRE	IRQ_2 E	IRQ_1 E	$0014

Bit 0, Bit 1 $\overline{IRQ_1}$, $\overline{IRQ_2}$ Enable Bit (IRQ_1E, IRQ_2E)

When using P_{50} and P_{51} as interrupt pins, write "1" in these bits. When the bit is set to "1", the DDRs corresponding to P_{50} and P_{51} are cleared and become $\overline{IRQ_1}$ input pin and $\overline{IRQ_2}$ input pin. When IRQ_1E and IRQ_2E are set, P_{50} and P_{51} cannot be used as an output ports. When "0", the CPU doesn't accept an external interrupt or a sleep cancellation by the external interrupt. These bits are cleared during reset.

Bit 2 Memory Ready Enable Bit (MRE)

When using P_{52} as an input pin of the "memory ready" signal, write "1" in this bit. When set, P_{52} DDR is automatically cleared and becomes the MR input pin. The bit is cleared during reset.

Bit 3 Halt Enable Bit (HLTE)

When using P_{53} as an input pin of the $\overline{HALT}$ signal, write "1" in this bit. When this bit is set, P_{53} DDR is automatically cleared and becomes the Halt input pin. If the bit is "0", the Halt function is inhibited and P_{53} is used as an I/O port. The bit is set to "1" during reset.

Bit 4 Auto Memory Ready Enable Bit (AMRE)

When the bit is set and the CPU accesses the external address, "memory ready" operates automatically and stretches the E clock's "High" duration for one system clock. When MRE bit of bit 2 is cleared and when the CPU accesses the external address space, the function operates. When MRE bit is set and then the CPU accesses the external address space with P_{52}(MR) pin in "low", "memory ready" operates automatically. This bit is set to "1" during reset.

Table 3 "Memory Ready" Function

MRE	AMRE	Function
0	0	"Memory ready" inhibited.
0	1	When the CPU accesses the external address, "High" duration of E clock automatically becomes one-cycle longer. This state is retained during reset.
1	0	"Memory ready" operates by P_{52} (MR) pin. The function is the same as that of the HD6301X0.
1	1	When the CPU accesses the external address space with the P_{52} (MR) pin in "low", the "auto memory ready" operates. This function is effective if it has both "high-speed memory" and "slow memory" outside. Input $\overline{CS}$ signal of "slow memory" to MR pin.

Bit 5 Standby Flag (STBY FLAG)

By clearing this flag, HD6303Y gets into the standby mode by software. This flag is set to "1" during reset, so the standby mode is canceled with $\overline{RES}$ pin in "low". The $\overline{RES}$ pin should be in "low" until oscillation becomes stable (min. 20ms.). If the $\overline{STBY}$ pin in is in "low", the standby mode can not be canceled with the $\overline{RES}$ pin in "low".

Bit 6 RAM Enable (RAME)

On-chip RAM can be disabled by this control bit. By resetting the MPU, "1" is set to this bit, and on-chip RAM is enabled. When this bit is cleared (=logic "0") on-chip RAM is invalid and the CPU can read data from external memory. This bit should be "0" before getting into the standby mode to protect on-chip RAM data.

Bit 7 Standby Power Bit (STBY PWR)

When V_{CC} is not provided in standby mode, this bit is cleared. This is a flag for read/write and can be read by software. If this bit is set before standyby mode, and remains set even after returning from standby mode, V_{CC} voltage is provided during standby mode and the on-chip RAM data is valid.

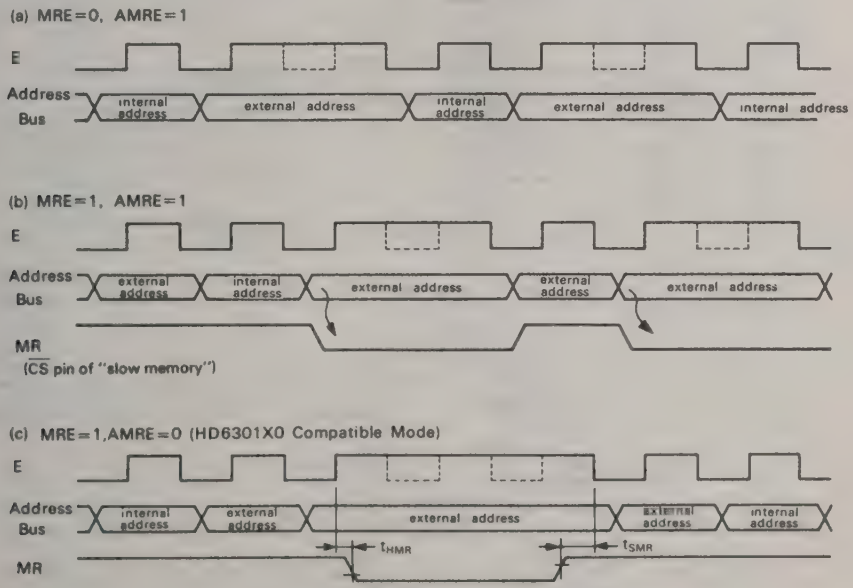


Figure 17 Memory Ready Timing

■ Port 6 Control/Status Register

This is the Control/Status Register for parallel handshake interface using Port 6. The functions are as follows;

1) Latches input data to Port 6 at the $\overline{IS}$ (P_{64}) falling edge.
2) Outputs a strobe signal $\overline{OS}$ (P_{65}) outward by reading or writing to port 6.
3) When IS FLAG is set at the $\overline{IS}$ falling edge, an interrupt occurs.

The following shows Port 6 Control/Status Register (P6CSR).

7	6	5	4	3	2	1	0	
IS* FLAG	IS IRQ1 ENABLE	OSE	OSS	LATCH ENABLE	–	–	–	$0021

*Bit 7 is Read-Only bit

Bit 0
Bit 1 Not used.
Bit 2

Bit 3: Latch Enable
This register controls the input latch for Port 6 (ISLATCH). When this bit is set to "1", the input data to port 6 will be latched inward at the $\overline{IS}$ (P_{54}) falling edge. An input latch will be canceled by reading Port 6, which enables to latch the next data. If cleared, the input latch remains canceled and this bit functions as a usual input port. This bit is cleared during reset.

Bit 4: OSS Output Strobe Select
This register initiates an output strobe ($\overline{OS}$) from P_{55} by reading or writing to port 6. When cleared, $\overline{OS}$ occurs by reading Port 6. When set, $\overline{OS}$ occurs by writing to Port 6. This bit is cleared during reset.

Bit 5: OSE Output Strobe Enable
This register decides the enabling or disabling of the output strobe. When cleared, P_{55} functions as an I/O port. When set, P_{55} functions as an $\overline{OS}$ output pin. (P_{55} DDR is set by OSE.) This bit is cleared during reset.

Bit 6: IS IRQ_1 Enable Input Strobe Interrupt Enable
When set, an $\overline{IRQ_1}$ interrupt to the CPU occurs by setting IS FLAG of bit 7. When cleared, the interrupt does not occur. This bit is cleared during reset.

Bit 7: IS Flag Input Strobe Flag
This flag is set at the IS (P_{54}) falling edge. This flag is for read-only. When set, the flag is cleared by reading or writing to Port 6 after reading the Port 6 Control Status Register. This bit is cleared during reset.

■ MEMORY MAP

The MPU can address up to 65k bytes. Memory map is shown in Fig. 20. 40 addresses ($0000 ~ $0027 except $00, $02, $04, $05, $06, $07, $18) are the internal registers as shown in Table 4.

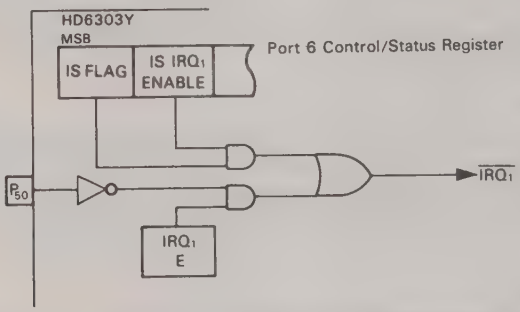


Figure 18 Input Strobe Interrupt block Diagram

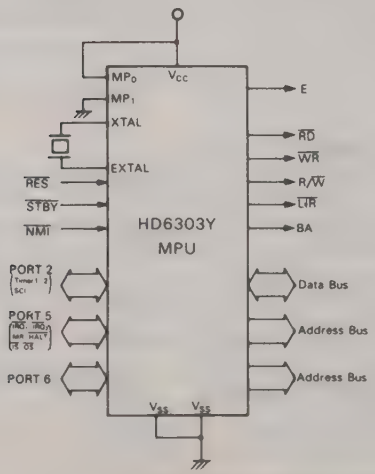


Figure 19 HD6303Y Operating Function

Table 4 Internal Register

Address	Register	Abbreviation	R/W**	Initialized value during reset***
00*	Port 1 DDR (Data Direction Register)	P1DDR	W	$FE
01	Port 2 DDR	P2DDR	W	$00
02*	Port 1	PORT1	R/W	indefinite
03	Port 2	PORT2	R/W	indefinite
04*	Port 3 DDR	P3DDR	W	$FE
05*	Port 4 DDR	P4DDR	W	$00
06*	Port 3	PORT3	R/W	indefinite
07*	Port 4	PORT4	R/W	indefinite
08	Timer Control/Status Register 1	TCSR1	R/W	$00
09	Free Running Counter (MSB)	FRCH	R/W	$00
0A	Free Running Counter (LSB)	FRCL	R/W	$00
0B	Output Compare Register 1 (MSB)	OCR1H	R/W	$FF
0C	Output Compare Register 1 (LSB)	OCR1L	R/W	$FF
0D	Input Capture Register (MSB)	ICRH	R	$00
0E	Input Capture Register (LSB)	ICRL	R	$00
0F	Timer Control/Status Register 2	TCSR2	R/W	$10
10	Rate/Mode Control Register	RMCR	R/W	$C0
11	Tx/Rx Control Status Register 1	TRCSR1	R/W	$20
12	Receive Data Register	RDR	R	$00
13	Transmit Data Register	TDR	W	indefinite
14	RAM/Port 5 Control Register	RP5CR	R/W	$F8 or $78
15	Port 5	PORT5	R/W	indefinite
16	Port 6 DDR	P6DDR	W	$00
17	Port 6	PORT6	R/W	indefinite
18	Port 7	PORT7	R/W	indefinite
19	Output Compare Register 2 (MSB)	OCR2H	R/W	$FF
1A	Output Compare Register 2 (LSB)	OCR2L	R/W	$FF
1B	Timer Control/Status Register 3	TCSR3	R/W	$20
1C	Time Constant Register	TCONR	W	$FF
1D	Timer 2 Up Counter	T2CNT	R/W	$00
1E	Tx/Rx Control Status Register 2	TRCSR2	R/W	$28
1F****	Test Register*	TSTREG	—	—
20	PORT 5 DDR	P5DDR	W	$00
21	PORT 6 Control/Status Register	P6CSR	R/W	$07
22	—	—	—	—
23	—	—	—	—
24	— Reserved	—	—	—
25	—	—	—	—
26	—	—	—	—
27	—	—	—	—

* External address.
** R: Read-only register, W: Write-only register, R/W: Read/Write register.
*** When empty bit is in the register, it is set to "1"
**** Register for test. Don't access this register.

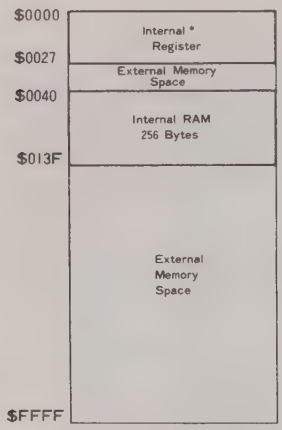


Figure 20 HD6303Y Memory Map

■ TIMER 1

The HD6303Y provides a 16-bit programmable timer which can simultaneously measure an input waveform and generate two independent output waveforms. The pulse widths of both input/output waveforms vary from microseconds to seconds.

Timer 1 is configured as follows (refer to Fig. 22).

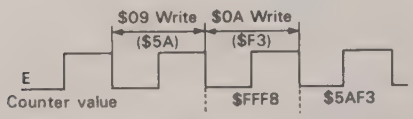
- Control/Status Register 1 (8 bit)
- Control/Status Register 2 (7 bit)
- Free Running Counter (16 bit)
- Output Compare Register 1 (16 bit)
- Output Compare Register 2 (16 bit)
- Input Capture Register (16 bit)

• Free-Running Counter (FRC)($0009:000A)

The key timer element is a 16-bit free-running counter driven and incremented by system clock. The counter value is readable by software without affecting the counter. The counter is cleared during reset.

When writing to the upper byte ($09), the CPU writes the preset value ($FFF8) into the counter (address $09, $0A) regardless of the write data value. But when writing to the lower byte ($0A) after the upper byte writing, the CPU writes not only lower byte data into lower 8 bit, but also upper byte data into higher 8 bit of the FRC.

The counter will be as follows when the CPU writes to it by double store instructions (STD, STX, etc.)



In the case of the CPU write ($5AF3) to the FRC

Figure 21 Counter Write Timing

• Output Compare Register (OCR) ($000B, $000C; OCR1) ($0019, $001A: OCR2)

The output compare register is a 16-bit read/write register which can control an output waveform. The data of OCR is always compared with the FRC.

When the data matches, output compare flag (OCF) in the timer control/status register (TCSR) is set. If an output enable bit (OE) in the TCSR2 is "1", an output level bit(OLVL) in the TCSR will be output to bit 1 (OCR 1) and bit 5 (OCR 2) of port 2. To control the output level again by the next compare, the value of OCR and OLVL should be changed. The OCR is set to $FFFF at reset. The compare function is inhibited for a cycle just after a write to the upper byte of the OCR or FRC. This is to set the 16-bit value valid in the counter register for compare. In addition, it is because counter is to set $FFF8 at the next cycle of the CPU's upper byte write to the FRC.

* For data write to the FRC or the OCR, 2-byte transfer instruction (such as STX, etc.) should be used.

• Input Capture Register (ICR) ($000D : 000E)

The input capture register is a 16-bit read-only register which stores the FRC's value when external input signal transition generates an input capture pulse. Such transition is controled by input edge bit (IEDG) in the TCSR1.

In order to input the external input signal to the edge detector, a bit of the DDR corresponding to bit 0 of port 2 should be cleared ("0"). When an input capture pulse occurs by external input signal transition at the next cycle of CPU's high-byte read of the ICR, the input capture pulse will be delayed by one cycle. In order to ensure the input capture operation, a CPU read of the ICR needs 2-byte transfer instruction. The input pulse width should be at least 2 system cycles. This register is cleared ($0000) during reset.

• Timer Control/Status Register 1 (TCSR1) ($0008)

The timer control/status register 1 is an 8-bit register. All bits are readable and the lower 5 bits are also writable. The upper 3 bits are read-only which indicate the following timer status.

Bit 5 The counter value reached to $0000 as a result of counting-up (TOF).

Bit 6 A match has occurred between the FRC and the OCR 1 (OCF1).

Bit 7 Defined transition of the timer input signal causes the counter to transfer its data to the ICR (ICF).

The followings are the each bit descriptions.

Timer Control/Status Register 1

7	6	5	4	3	2	1	0	
ICF	OCF1	TOF	EICI	EOCI1	ETOI	IEDG	OLVL1	$0008

Bit 0 OLVL1 Output Level 1

OLVL1 is transferred to port 2, bit 1 when a match occurs between the counter and the OCR1. If bit 0 of the TCSR2 (OE1), is set to "1", OLVL1 will appear at bit 1 of port 2.

Bit 1 IEDG Input Edge

This bit determines which edge, rising or falling, of input signal of bit 0 of port 2 will trigger data transfer from the counter to the ICR. For this function, the DDR corresponding to port 2, bit 0 should be cleared beforehand.

IEDG=0, triggered on a falling edge ("High" to "Low")

IEDG=1, triggered on a rising edge ("Low" to "High")

Bit 2 ETOI Enable Timer Overflow Interrupt

When this bit is set, an internal interrupt (IRQ_3) by TOI interrupt is enabled. When cleared, the interrupt is inhibited.

Bit 3 EOCI1 Enable Output Compare Interrupt 1

When this bit is set, an internal interrupt (IRQ_2) by OCI1 interrupt is enabled. When cleared, the interrupt is inhibited.

Bit 4 EICI Enable Input Capture Interrupt

When this bit is set, an internal interrupt (IRQ_2) by ICI interrupt is enabled. When cleared, the interrupt is inhibited.

Bit 5 TOF Timer Overflow Flag

This read-only bit is set when the counter increments from $FFFF by 1. Cleared when the counter's MSB byte ($0009) is read by the CPU after the TCSR1 read at TOF=1.

Bit 6 OCF1 Output Compare Flag 1

This read-only bit is set when a match occurs between the OCR1 and the FRC. Cleared when writing to the OCR1 ($000B or $000C) after the TCSR1 or TCSR2 read at OCF=1.

Bit 7 ICF Input Capture Flag

This read-only bit is set when an input signal of port 2, bit 0 makes a transition as defined by IEDG and the FRC is transferred to the ICR. Cleared when reading the upper byte ($000D) of the ICR after the TCSR1 or TCSR2 read at ICF=1.

- **Timer Control/Status Register 2 (TCSR2) ($000F)**

The timer control/status register 2 is a 7-bit register. All bits are readable and the lower 4 bits are also writable. But the upper 3 bits are read-only which indicate the following timer status.

Bit 5 A match has occurred between the FRC and the OCR2 (OCF2).

Bit 6 The same status flag as the OCF1 flag of the TCSR1, bit 6.

Timer Control/Status Register 2

7	6	5	4	3	2	1	0	
ICF	OCF1	OCF2	–	EOCI2	OLVL2	OE2	OE1	$000F

Bit 7 The same status flag as the ICF flag of the TCSR1, bit 7.

The followings are the each bit descriptions.

Bit 0 OE1 Output Enable 1

This bit enables the OLVL1 to appear at port 2, bit 1 when a match has occurred between the counter and the output compare register 1. When this bit is cleared, bit 1 of port 2 will be an I/O port. When set, it will be an output of OLVL1 automatically.

Bit 1 OE2 Output Enable 2

This bit enables the OLVL2 to appear at port 2, bit 5 when a match has occurred between the counter and the output compare register 2. When this bit is cleared, port 2, bit 5 will be an I/O port. When set, it will be an output of OLVL2 automatically.

Bit 2 OLVL2 Output Level 2

OLVL2 is transferred to port 2, bit 5 when a match has occurred between the counter and the OCR2. If bit 5 of the TCSR2 (OE2), is set to "1", OLVL2 will appear at port 2, bit 5.

Bit 3 EOCI2 Enable Output Compare Interrupt 2

When this bit is set, an internal interrupt (IRQ_2) by OCI2 interrupt is enabled. When cleared, the interrupt is inhibited.

Bit 4 Not used

Bit 5 OCF2 Output Compare Flag 2

This read-only bit is set when a match has occurred between the counter and the OCR2. Cleared when writing to the OCR2 ($0019 or $001A) after the TCSR2 read at OCF2=1.

Bit 6 OCF1 Output Compare Flag 1

Bit 7 ICF Input Capture Flag

OCF1 and ICF are dual addressed. If which register, TCSR1 or TCSR2, CPU reads, it can read OCF1 and ICF to bit 6 and bit 7.

Both the TCSR1 and TCSR2 will be cleared during reset.

(Note) If OE1 or OE2 is set to "1" before the first output compare match occurs after reset restart, bit 1 or bit 5 of port 2 will produce "0" respectively.

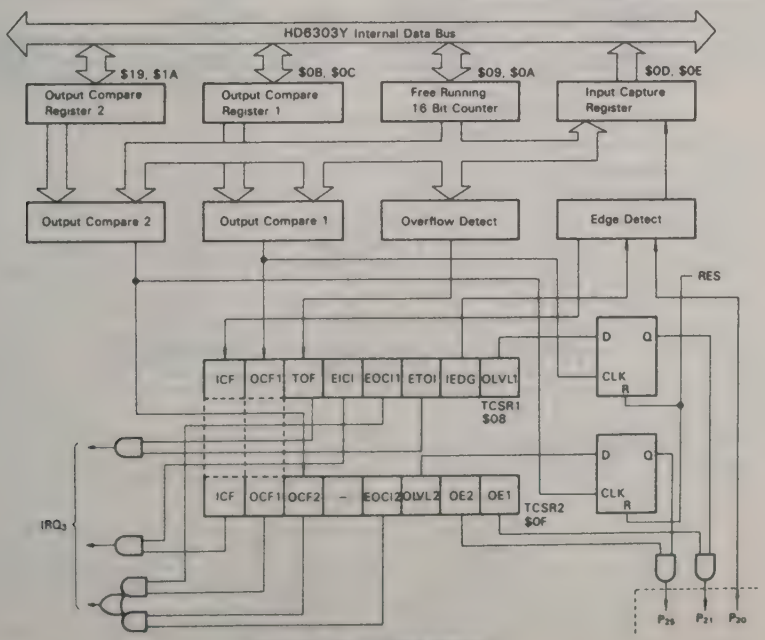


Figure 22 Timer 1 Block Diagram

■ TIMER 2

In addition to the timer 1, the HD6303Y provides an 8-bit reloadable timer, which is capable of counting the external event. The timer 2 contains a timer output, so the MPU can generate three independent waveforms. (Refer to Fig. 23.)

The timer 2 is configured as follows:

- Control/Status Register 3 (7 bits)
- 8-bit Up Counter
- Time Constant Register (8 bits)

• Timer 2 Up Counter (T2CNT) ($001D)

This is an 8-bit up counter which operates with the clock decided by CKS0 and CKS1 of the TCSR3. The CPU can read the value of the counter without affecting the counter. In addition, any value can be written to the counter by software even during counting.

The counter is cleared when a match occurs between the counter and the TCONR or during reset.

If the write operation is made by software to the counter at the cycle of counter clear, it does not reset the counter but put the write data to the counter.

• Time Constant Register (TCONR) ($001C)

The time constant register is an 8-bit write only register. The data of register is always compared with the counter.

When a match has occurred, the counter match flag (CMF) of the timer control status register 3 (TCSR3) is set and the value selected by TOS0 and TOS1 of the TCSR3 will appear at port 2, bit 6. When CMF is set, the counter will be cleared simultaneously and then start counting from $00. This enables regular interrupts and waveform outputs without any software support. The TCONR is set to "$FF" during reset.

• Timer Control/Status Register 3 (TCSR3) ($001B)

The timer control/status register 3 is a 7-bit register. All bits are readable and 6 bits except for CMF can be written.

The followings are each pin descriptions.

Timer Control/Status Register 3

7	6	5	4	3	2	1	0	
CMF	ECMI	–	T2E	TOS1	TOS0	CKS1	CKS0	$001B

Bit 0 CKS0 Input Clock Select 0
Bit 1 CKS1 Input Clock Select 1

Input clock to the counter is selected as shown in Table 5 depending on these two bits. When an external clock is selected, bit 7 of port 2 will be a clock input automatically. Timer 2 detects the rising edge of the external clock and increments the counter. The external clock is countable up to half the frequency of the system clock.

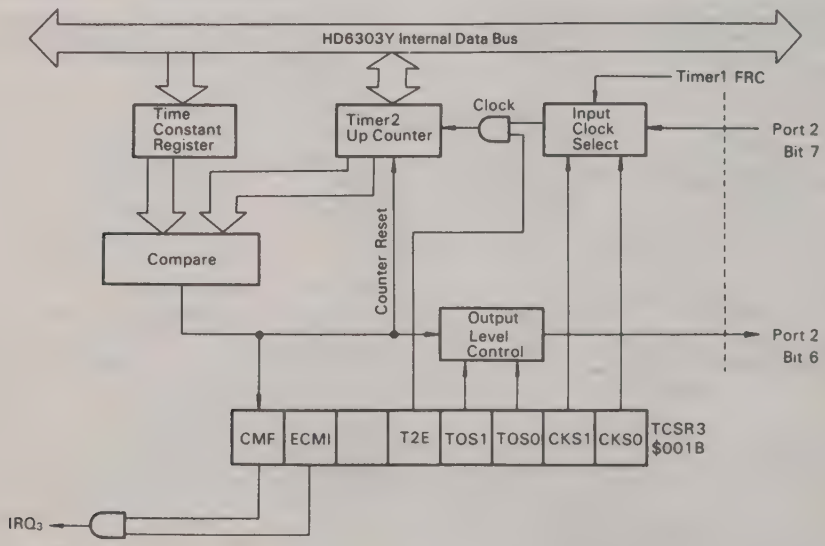


Figure 23 Timer 2 Block Diagram

Table 5 Input Clock Select

CKS1	CKS0	Input Clock to the Counter
0	0	E clock
0	1	E clock/8*
1	0	E clock/128*
1	1	External clock

* These clocks come from the FRC of the timer 1. If one of these clocks is selected as an input clock to the up counter, the CPU should not write to the FRC of the timer 1.

Bit 2 TOS0 Timer Output Select 0
Bit 3 TOS1 Timer Output Select 1

When a match occurs between the counter and the TCONR timer 2 outputs shown in Table 6 will appear at port 2, bit 6 depending on these two bits. When both TOS0 and TOS1 are "0", bit 6 of port 2 will be an I/O port.

Table 6 Timer 2 Output Select

TOS1	TOS0	Timer Output
0	0	Timer Output Inhibited
0	1	Toggle Output*
1	0	Output "0"
1	1	Output "1"

* When a match occurs between the counter and the TCONR, timer 2 output level is reversed. This leads to production of a square wave with 50% duty to the external without any software support.

Bit 4 T2E Timer 2 Enable Bit

When this bit is cleared, a clock input to the up counter is inhibited and the up counter stops. When set to "1", a clock selected by CKS1 and CKS0 (Table 5) is input to the up counter.

(Note) P_{26} outputs "0" when T2E bit cleared and timer 2 set in output enable condition by TOS1 or TOS0. It also outputs "0" when T2E bit set "1" and timer 2 set in output enable condition before the first counter match occurs.

Bit 5 Not Used.

Bit 6 ECMI Enable Counter Match Interrupt

When this bit is set, an internal interrupt (IRQ_3) by CMI is enabled. When cleared, the interrupt is inhibited.

Bit 7 CMF Counter Match Flag

This read-only bit is set when a match occurs between the up counter and the TCONR. Cleared by writing "0" at CMF=1 by software (unable to write "1" by software).

Each bit of the TCSR3 is cleared during reset.

■ SERIAL COMMUNICATION INTERFACE (SCI)

The Serial Communication Interface (SCI) in the HD6303Y contains the following two operating modes: asynchronous mode by the NRZ format, and clocked synchronous mode which transfers data synchronously with the clock. In the asynchronous mode, data length, parity bits and number of stop bits can be selected, and eight transfer formats are provided.

The SCI consists of the following registers as shown in Fig. 24 Block Diagram.

- Transmit/Receive Control Status Register 1 (TRCSR1)
- Rate/Mode Control Register (RMCR)
- Transmit/Receive Control Status Register 2 (TRCSR2)
- Receive Data Register (RDR)
- Recevie Shift Register
- Transmit Data Register (TDR)
- Transmit Shift Register

To operate the SCI, initialize the RMCR and TRCSR2, after selecting the desirable operating mode and transfer format. Next, set the enable bit (TE or RE) of the TRCSR1. Operating mode and transfer format should be changed when the enable bit (TE, RE) is cleared. When setting the TE or RE again after changing the operating mode or transfer format, interval of more than a 1-bit cycle of the baud rate or bit rate is necessary. If a 1-bit cycle or more is not allowed, the SCI block may not be initialized.

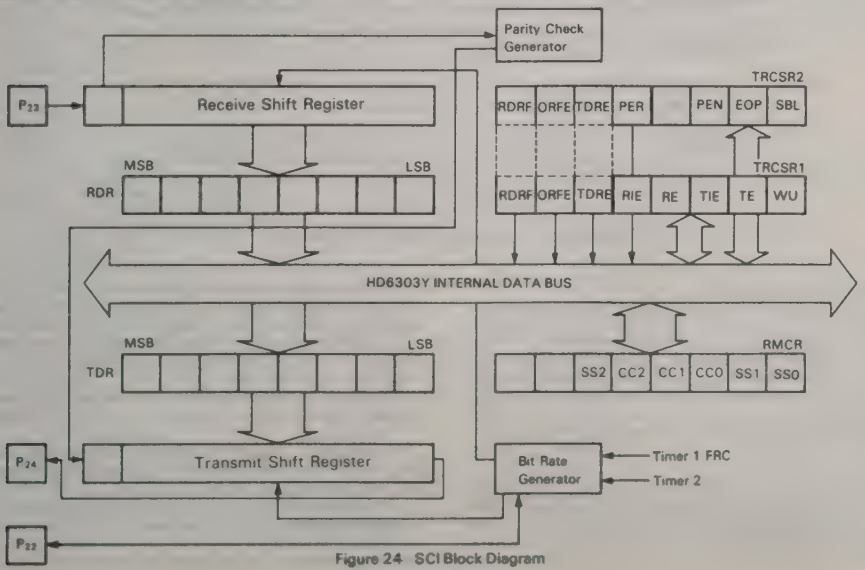


Figure 24 SCI Block Diagram

- **Asynchronous Mode**

Asynchronous mode contains 8 transfer formats as shown in Fig. 25.

Data transmission is enabled by setting TE bit of the TRCSR1, then port 2, bit 4 will unconditionally become a serial output independently of the corresponding DDR.

To transmit data, set the desirable transmit format with RMCR and TRCSR2. When the TE bit is set, the data can be transmitted after transmitting the one frame of preamble ("1").

The conditions at this stage are as follows.

1) If the TDR is empty (TDRE=1), consecutive 1's are produced to indicate the idle state.
2) If the TDR contains data (TDRE=0), data is sent to the Transmit Shift Register and data transmit starts.

During data transmit, a start bit of "0" is transmitted first. Then 7-bit or 8-bit data (starts from bit 0) is transmitted. With PEN=1, the parity bit, even or odd, selected by EOP bit is added, lastly the stop bit (1 bit or 2 bis) is sent.

When the TDR is "empty", hardware sets TDRE flag bit. If the CPU doesn't respond to the flag in proper timing (the TDRE is in set condition till the next normal data transfer starts from the transmit data register to the transmit sift register), "1" is transferred instead of the start bit "0" and continues to be transferred till data is provided to the data register. While the TDRE is "1", "0" is not transferred.

Data receive is possible by setting RE bit. This makes port 2, bit 3 a serial input. The operation mode of data receive is decided by the contents of the TRCSR2 and RMCR at first, and set RE bit of TRCSR1. The first "0" (space) synchronizes the receive bit flow. Each bit of the following data will be strobed in the middle. If a stop bit is not "1", a framing error assumed and ORFE is set.

When a framing error occurs, receive data is transferred to the Receive Data Register and the CPU can read the error-generating data. This makes it possible to detect a line break.

When PEN bit is set, the parity check is done. If the parity bit does not match the EOP bit, a parity error occurs and the PER bit is set, not the RDRF bit. Also, when the parity error occurs the receive data can be read just like in the case of the framing error.

The RDRF flag is set when the data is received without a framing error and a parity error.

If RDRF is still set when receiving the stop bit of the next data, ORFE is set to indicate the overrun generation. CPU can get the receive data by reading RDR. When 7 bit data format is selected, the 8th bit of RDR is "0".

When the CPU read the receive Data Register as a response to RDRF flag or ORFE flag after having read TRCSR, RDRF or ORFE is cleared.

(Note) Clock Source in Asynchronous Mode

If CC1:CC0=10, the internal bit rate clock is provided at P_{22} regardless of the values for TE or RE. Maximum clock rate is E÷16.

If both CC1 and CC0 are set, an external TTL compatible clock must be connected to P_{22} at sixteen times (16×) the desired bit rate, but not greater than E.

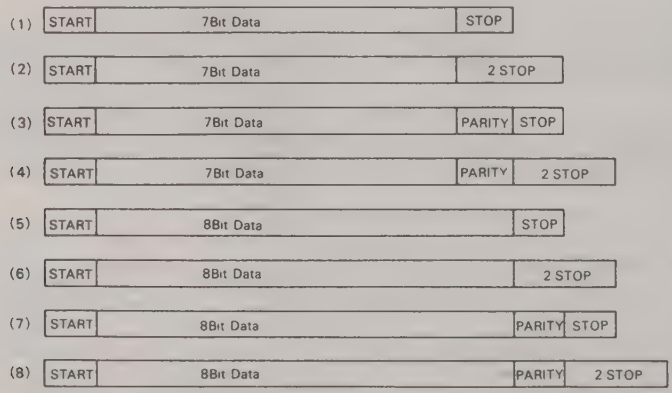

(1) START 7Bit Data STOP
(2) START 7Bit Data 2 STOP
(3) START 7Bit Data PARITY STOP
(4) START 7Bit Data PARITY 2 STOP
(5) START 8Bit Data STOP
(6) START 8Bit Data 2 STOP
(7) START 8Bit Data PARITY STOP
(8) START 8Bit Data PARITY 2 STOP

Figure 25 Asynchronous Mode Transfer Format

- **Clocked Synchronous Mode**

In the clocked synchronous mode, data transmit is synchronized with the clock pulse. The HD6303Y SCI provides functionally independent transmitter and receiver which makes full duplex operation possible in the asynchronous mode. But in the clocked synchronous mode an SCI clock I/O pin is only P_{22}, so the simultaneous receive and transmit operation is not available. In this mode, TE and RE should not be in set condition ("1") simultaneously. Fig. 26 gives a synchronous clock and a data format in the clocked synchronous mode.

1) Data transmit

Data transmit is realized by setting TE bit in the TRCSR1. Port 2, bit 4 becomes an output unconditionally independent of the value of the corresponding DDR.

Both the RMCR and TRCSR should be set in the desirable operating condition for data transmit.

When an external clock input is selected and the TDRE flag is "0", data transmit is performed from port 2, bit 4, synchronizing with 8 clock pulses input from external to port 2, bit 2.

Data is transmitted from bit 0 and the TDRE is set when the Transmit Shift Register (TSR) is "empty". More than 9th clock pulse of external are ignored.

When data transmit is selected to the clock output, the MPU produces transmit data and synchronous clock at TDRE flag clear.

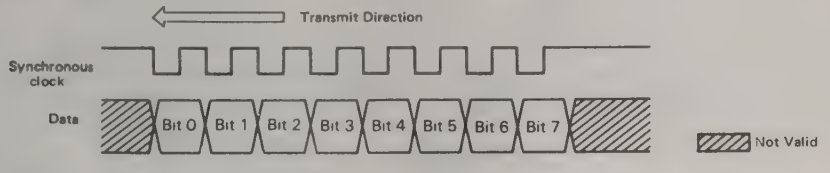
2) Data receive

Data receive is enabled by setting RE bit. Port 2, bit 3 will be a serial input. The operating mode of data receive is decided by the TRCSR1 and the RMCR.

If the external clock input is selected, 8 external clock pulses and the synchronized receive data are input to port 2, bit 2 and bit 3 respectively. The MPU put receive data into the receive data shift register by this clock and set the RDRF flag at the termination of 8 bit

data receive. More than 9th clock pulse of external input are ignored. When RDRF is cleared, the MPU starts receiving the next data instantly. So, RDRF should be cleared with P_{22} "High".

When data receive is selected with the clock output, 8 synchronous clocks are output to the external by setting RE bit. So receive data should be input from external synchronously with this clock. When the first byte data is received, the RDRF flag is set. After the second byte, receive operation is performed by sending the synchronous clock to the external after clearing the RDRF bit.



- Transmit data is produced from a falling edge of a synchronous clock to the next falling edge.
- Receive data is latched at the rising edge.

Figure 26 Clocked Synchronous Mode Format

• Transmit/Receive Control Status Register (TRCSR1) ($0011)

The TRCSR1 is composed of 8 bits which are all readable. Bits 0 to 4 are also writable. This register is initialized to $20 during reset. Each bit functions are as follows.

Transmit/Receive Control Status Register

7	6	5	4	3	2	1	0	
RDRF	ORFE	TDRE	RIE	RE	TIE	TE	WU	$0011

Bit 0 WU Wake-up

In a typical multi-processor configuration, the software protocol provides the destination address at the first byte of the message. In order to make uninterested MPU ignore the remaining message, a wake-up function is available. By this, uninterested MPU can inhibit all further receive processing till the next message starts.

Then wake-up function is triggered by consecutive 1's with 1 frame length. The software protocol should provide the idle time between messages.

By setting this bit, the MPU stops data receive till the next message. The receive of consecutive "1" with one frame length wakes up and clears this bit by hardware and then the MPU restarts receive operation. However, the RE flag should be already set before setting this bit. In the clocked synchronous mode WU is not available, so this bit should not be set.

Bit 1 TE Transmit Enable

When this bit is set, transmit data will appear at port 2, bit 4 after one frame preamble in asynchronous mode, while in clocked synchronous mode it appears immediately. This is executed regardless of the value of the corresponding DDR. When TE is cleared, the serial I/O doesn't affect port 2, bit 4.

Bit 2 TIE Transmit Interrupt Enable

When this bit is set, an internal interrupt (IRQ_3) is enabled when TDRE (bit 5) is set. When cleared, the interrupt is inhibited.

Bit 3 RE Receive Enable

When set, a signal is input to the receiver from port 2, bit 3 regardless of the value of the DDR. When RE is cleared, the serial I/O doesn't afffect port 2, bit 3.

Bit 4 RIE Receive Interrupt Enable

When this bit is set, an internal interrupt (IRQ_3) is enabled when RDRF (bit 7) or ORFE (bit 6) is set. When cleared, the interrupt is inhibited.

Bit 5 TDRE Transmit Data Register Empty

TDRE is set by hardware when the TDR is transferred to the Transmit Shift Register in the asynchronous mode, while in clocked synchronous mode when the TDSR is "empty". This bit is cleared by reading the TRCSR1 or TRCSR2 and writing new transmit data to the TDR when TDRE=1 TDRE is set to "1" during reset.

Bit 6 ORFE Overrun Framing Error

ORFE is set by hardware when an overrun or a framing error is generated (during data-receive only). An overrun error occurs when new receive data is ready to be transferred to the RDR during RDRF still being set. A framing error occurs when a stop bit is "0". But in clocked synchronous mode, this bit is not affected. This bit is cleared by reading the TRCSR1 or TRCSR2, and the RDR, when RDRF=1. ORFE is cleared during reset.

Bit 7 RDRF Receive Data Register Full

RDRF is set by hardware when data is received normally and transferred from the Receive Shift Register (RSR) to the RDR. This bit is cleared by reading TRCSR1 or TRCSR2, and the RDR, when RDRF=1. This bit is cleared during reset.

• Transmit Rate/Mode Control Register (RMCR)

The RMCR controls the following serial I/O:

- Baud Rate
- Clock source
- Operation Mode
- Data Format
- Port 2, Bit 2 Function

All bits are readable/writable. Bit 0 to 5 of the RMCR are cleared during reset.

Transfer Rate/Mode Control Register

7	6	5	4	3	2	1	0	
–	–	SS2	CC2	CC1	CC0	SS1	SS0	$0010

Bit 0 SS0
Bit 1 SS1 Speed Select
Bit 5 SS2

These bits control the baud rate used for the SCI. Table 7 lists the available baud rates. The timer 1 FRC (SS2=0) and the timer 2 up counter (SS2=1) provide the internal clock to the SCI. When selecting the timer 2 as a baud rate clock source, it functions as a baud rate generator. The timer 2 generates the baud rate listed in Table 8 depending on the value of the TCONR.

(Note) When operating the SCI with internal clock, do not perform write operation to the timer/counter which is the

Table 7 SCI Bit Times and Transfer Rates

(1) Asynchronous Mode

			XTAL	2.4576MHz	4.0MHz	4.9152MHz
SS2	SS1	SS0	E	614.4kHz	1.0MHz	1.2288MHz
0	0	0	E÷16	26 μs/38400Baud	16 μs/62500Baud	13 μs/76800Baud
0	0	1	E÷128	208 μs/4800Baud	128 μs/7812.5Baud	104.2 μs/9600Baud
0	1	0	E÷1024	1.67ms/600Baud	1.024ms/976.6Baud	833.3 μs/1200Baud
0	1	1	E÷4096	6.67ms/150Baud	4.096ms/244.1Baud	3.333ms/300Baud
1	—	—	—	*	*	*

* When SS2 is "1", Timer 2 provides SCI clocks. The baud rate is shown as follows with the TCONR as N.

$$\text{Baud Rate} = \frac{f}{32(N+1)} \qquad \begin{pmatrix} f: \text{input clock frequency to the timer 2 counter} \\ N = 0 \sim 255 \end{pmatrix}$$

(2) Clocked Synchronous Mode *

			XTAL	4.0MHz	6.0MHz	8.0MHz
SS2	SS1	SS0	E	1.0MHz	1.5MHz	2.0MHz
0	0	0	E÷2	2 μs/bit	1.33 μs/bit	1 μs/bit
0	0	1	E÷16	16 μs/bit	10.7 μs/bit	8 μs/bit
0	1	0	E÷128	128 μs/bit	85.3 μs/bit	64 μs/bit
0	1	1	E÷512	512 μs/bit	341 μs/bit	256 μs/bit
1	—	—	—	**	**	**

* Bit rates in the case of internal clock operation. In the case of external clock operation, the external clock is operatable up to DC ~ 1/2 system clock.

** The bit rate is shown as follows with the TCONR as N.

$$\text{Bit Rate } (\mu\text{s/bit}) = \frac{4(N+1)}{f} \qquad \begin{pmatrix} f: \text{input clock frequency to the timer 2 counter} \\ N = 0 \sim 255 \end{pmatrix}$$

Table 8 Baud Rate and Time Constant Register Example

Baud Rate (Baud) \ XTAL	2.4576MHz	3.6864MHz	4.0MHz	4.9152MHz	8.0MHz
110	21*	32*	35*	43*	70*
150	127	191	207	255	51*
300	63	95	103	127	207
600	31	47	51	63	103
1200	15	23	25	31	51
2400	7	11	12	15	25
4800	3	5	—	7	12
9600	1	2	—	3	—
19200	0	—	—	1	—
38400	—	—	—	0	—

* E/8 clock is input to the timer 2 up counter and E clock otherwise.

clock source of the SCI.

Bit 2 CC0
Bit 3 CC1 Clock Control/Format Select*
Bit 4 CC2

These bits control the data format and the clock source (refer to Table 9).

* CC0, CC1 and CC2 are cleared during reset and the MPU goes to the clocked synchronous mode of the external clock operation. Then the MPU automatically set port 2, bit 2 into the clock input state. When using port 2, bit 2 as an output port, the DDR of port 2 should be set to "1" and CC1 and CC0 to "0" and "1" respectively.

Bit 6 Not Used.
Bit 7 Not Used

• Transmit/Receive Control Status Register 2 (TRCSR2)

The TRCSR2 is a 7-bit register which can select a data format in the asynchronous mode. The upper 3 bits are the same address as the TRCSR1. Therefore, the RDRF, ORFE and TDRE can be read by either the TRCSR1 or TRCSR2. Bits 0 to 2 of the TRCSR2 are used for read/write. Bits 4 to 7 are used only for read.

Transmit/Receive Control Status Register 2

7	6	5	4	3	2	1	0	
RDRF	ORFE	TDRE	PER	—	PEN	EOP	SBL	$001E

Bit 0 SBL Stop Bit Length

This bit selects the stop bit length in the asynchronous mode. If this bit is "0", the stop bit is 1-bit. If "1", the stop bit is 2-bit. This bit is cleared during reset.

Bit 1 EOP Even/Odd Parity

This bit selects the parity generated and checked when the PEN is "1". If this bit is "0", the parity is even. If "1", it is odd. This bit is cleared during reset.

Bit 2 PEN Parity Enable

This bit decides whether the parity bit should be generated and checked in the asynchronous mode or not. If this bit is "0", the parity bit is neither generated nor checked. If "1", it is generated and checked. This bit is cleared during reset.

The 3 bits above do not affect the SCI opertion in the clocked synchronous mode.

Bit 3 Not Used

Bit 4 PER Parity Error

This bit is set when the PEN is "1" and a parity error occurs. It is cleared by reading the RDR after reading the TRCSR2, when PER=1.

Bit 5 TDRE

Transmit Data Register Empty

Bit 6 ORFE

Overrun/Framing Error

Bit 7 RDRF

Receive Data Register Full

* Each flag of the TDRE, ORFE, and RDRF can be read from either the TRCSR1 or TRCSR2.

■ TIMER, SCI STATUS FLAG

Table 10 shows the set and reset conditions of each status flag in the timer 1, timer 2 and SCI.

Table 9 SCI Format and Clock Source Control

CC2	CC1	CC0	Format	Mode	Clock Source	Port 2, Bit 2	Port 2, Bit 3	Port 2, Bit 4
0	0	0	8-bit data	Clocked Synchronous	External	Input	When the TRCSR1, RE bit is "1", bit 3 is used as a serial input.	
0	0	1	8-bit data	Asynchronous	Internal	Not Used**		
0	1	0	8-bit data	Asynchronous	Internal	Output*		
0	1	1	8-bit data	Asynchronous	External	Input		
1	0	0	8-bit data	Clocked Synchronous	Internal	Output	When the TRCSR1, TE bit is "1", bit 4 is used as a serial output.	
1	0	1	7-bit data	Asynchronous	Internal	Not Used**		
1	1	0	7-bit data	Asynchronous	Internal	Output*		
1	1	1	7-bit data	Asynchronous	External	Input		

* Clock output regardless of the TRCSR1, bit RE and TE.
** Not used for the SCI.

Table 10 Timer 1, Timer 2 and SCI Status Flag

		Set Condition	Clear Condition
P6CSR	IS FLAG	Falling edge input to P_{54} ($\overline{IS}$)	1. Read the P6CSR then read or write the PORT6, when IS FLAG = 1 2. $\overline{RES}$ = 0
Timer 1	ICF	FRC → ICR by Rising or Falling edge input to P_{20}. (Selecting with the IEDG bit)	1. Read the TCSR1 or TCSR2 then ICRH, when ICF = 1 2. $\overline{RES}$ = 0
	OCF1	OCR1 = FRC	1. Read the TCSR1 or TCSR2 then write to the OCR1H or OCR1L, when OCF1 = 1 2. $\overline{RES}$ = 0
	OCF2	OCR2 = FRC	1. Read the TCSR2 then write to the OCR2H or OCR2L, when OCF2 = 1 2. $\overline{RES}$ = 0
	TOF	FRC = $FFFF + 1 cycle	1. Read the TCSR1 then FRCH, when TOF = 1 2. $\overline{RES}$ = 0
Timer 2	CMF	T2CNT = TCONR	1. Write "0" to CMF, when CMF = 1 2. $\overline{RES}$ = 0
SCI	RDRF	Receive Shift Register → RDR	1. Read the TRCSR1 or TRCSR2 then RDR, when RDRF = 1 2. $\overline{RES}$ = 0
	ORFE	1. Framing Error (Asynchronous Mode) Stop Bit = 0 2. Overrun Error (Asynchronous Mode) Receive Shift Register → RDR when RDRF = 1	1. Read the TRCSR1 or TRCSR2 then RDR, when ORFE = 1 2. $\overline{RES}$ = 0
	TDRE	1. Asynchronous Mode TDR → Transmit Shift Register 2. Clocked Synchronous Mode Transmit Shift Register is "empty" 3. $\overline{RES}$ = 0	Read the TRCSR1 or TRCSR2 then write to the TDR, when TDRE = 1
	PER	Parity when PEN = 1	1. Read the TRCSR2 then RDR, when PER = 1 2. $\overline{RES}$ = 0

(Note) → ; Transfer = ; equal

ICRH; Upper byte of ICR
OCR1H; Upper byte of OCR1
OCR2H; Upper byte of OCR2

OCR1L; Lower byte of OCR1
OCR2L; Lower byte of OCR2
FRCH; Upper byte of FRC

■ LOW POWER DISSIPATION MODE

The HD6303Y provides two low power dissipation modes; sleep and standby.

• Sleep Mode

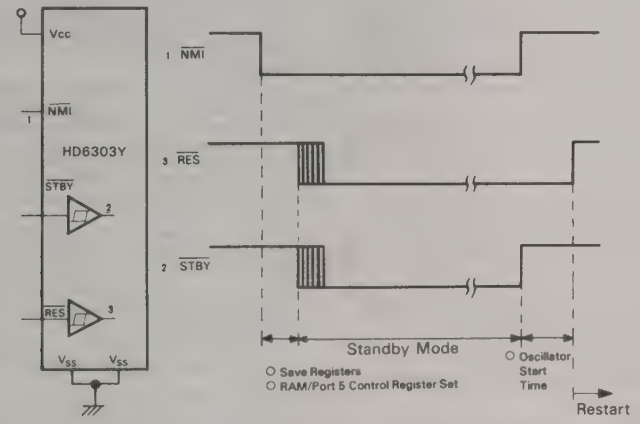
The MPU goes to the sleep mode by SLP instruction execution. In the sleep mode, the CPU stops its operation, while the registers' contents are retained. In this mode, the peripherals except the CPU such as timers, SCI, etc. continue their functions. The power dissipation of sleep-condition is one fourth that of operating condition.

The MPU returns from this mode by an interrupt, $\overline{RES}$ or $\overline{STBY}$; it goes to the reset state by $\overline{RES}$ and the standby mode by $\overline{STBY}$. When the CPU acknowledges an interrupt request, it cancels the sleep mode, returns to the operation mode and branches to the interrupt routine. When the CPU masks this interrupt, it cancels the sleep mode and executes the next instruction. However, for example, if the timer 1 or 2 prohibits a timer interrupt, the CPU doesn't cancel the sleep mode because of no interrupt request.

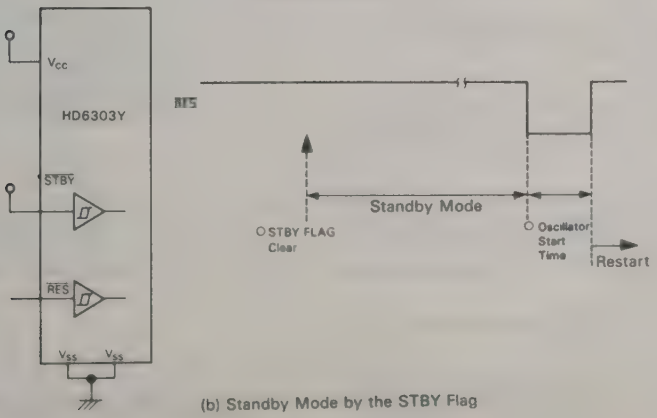
This sleep mode is effective to reduce the power dissipation for a system with no need of the HD6303Y's consecutive operation.

• Standby Mode

The MPU goes to the standby mode with the $\overline{STBY}$ "Low" or by clearing the STBY flag. In this mode, the HD6303Y stops all the clocks and goes to the reset state. In this mode, the power dissipation is reduced to several μA. During standby, all pins, except the power supply (V_{CC}, V_{SS}), the $\overline{STBY}$, $\overline{RES}$ and XTAL (which outputs "0"), go to the high impedance state. In this mode, power (V_{CC}) is supplied to the HD6303Y, and the contents of RAM is retained. The MPU returns from this mode during reset. When the MPU goes to the standby mode with $\overline{STBY}$ "Low", it will restart at the timing shown in Fig. 27(a). When the MPU goes to the standby mode by clearing the STBY flag, it will restart only by keeping the $\overline{RES}$ "Low" for longer than the oscillating stabilization time. (Fig. 27(b))



(a) Standby Mode by $\overline{STBY}$



(b) Standby Mode by the STBY Flag

Figure 27 Standby Mode Timing

■ TRAP FUNCTION

The CPU generates an interrupt with the highest priority (TRAP) when fetching an undefined instruction or an instruction from non-memory space. The TRAP prevents the system-burst caused by noise or a program error.

• Op Code Error

When fetching an undefined op code, the CPU saves registers as well as a normal interrupt and branches to the TRAP ($FFEE, $FFEF). This has the priority next to reset.

• Address Error

When an instruction fetch is made from the address of internal register, the MPU generaters an interrupt as well as an op code error. But on the system with no memory in its external memory area, this function is not applicable if an instruction fetch is made from the external non-memory area. Addresses where an address error occurs are from $0000 to $0027.

This function is available only for an instruction fetch and is not applicable to the access of normal data read/write.

(Note) The TRAP interrupt provides a retry function differently from other interrupts. This is a program flow return to the address where the TRAP occurs when a sequence returns to a main routine from the TRAP interrupt routine by RTI. The retry can prevent the system burst caused by noise, etc.
However, if another TRAP occurs, the program repeats the TRAP interrupt forever, so the consideration is necessary in programming.

■ INSTRUCTION SET

The HD6303Y provides object code upward compatible with the HD6801 to utilize all instruction set of the HMCS6800. It also reduces the execution times of key instructions for throughput improvement.

Bit manipulation instruction, change instruction of the index register and accumulator and sleep instruction are also added.

The followings are explained here.

- CPU Programming Model (refer to Fig. 28)
- Addressing Mode
- Accumulator and Memory Manipulation Instruction (refer to Table 11)
- New Instruction
- Index Register and Stack Manipulation Instruction (refer to Table 12)
- Jump and Branch Instruction (refer to Table 13)
- Condition Code Register Manipulation (refer to Table 14)
- Op Code Map (refer to Table 15)

• Programming Model

Fig. 28 depicts the HD6303Y programming model. The double accumulator D consists of accumulator A and B, so when using the accumulator D, the contents of A and B are destroyed.

• CPU Addressing Mode

The HD6303Y provides 7 addressing modes. The addressing mode is determined by an instruction type and code. Tables 11 through 15 show addressing modes of each instruction with the execution times counted by the machine cycle.

When the clock frequency is 4MHz, the machine cycle time becomes microseconds directly.

Accumulator (ACCX) Addressing

Only an accumulator is addressed and the accumulator A or B is selected. This is a one-byte instruction.

Immediate Addressing

This addressing locates a data in the second byte of an instruction. However, LDS and LDX locate a data in the second and third byte exceptionally. This addressing is a 2 or 3-byte instruction.

Direct Addressing

In this addressing mode, the second byte of an instruction shows the address where a data is stored. 256 bytes ($0 through $255) can be addressed directly. Execution times can be reduced by storing data in this area so it is recommended to make it RAM for users' data area in configurating a system. This is a 2-byte instruction, while 3 byte with regard to AIM, OIM, EIM and TIM.

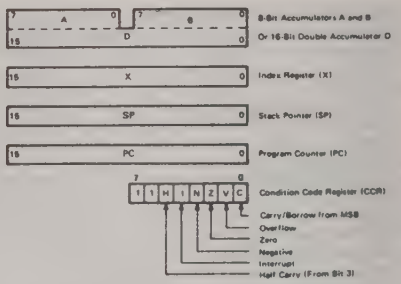


Figure 28 CPU Programming Model

Extended Addressing

In this mode, the second byte shows the upper 8 bit of the data stored address and the third byte the lower 8 bit. This indicates the absolute address of 3 byte instruction in the memory.

Indexed Addressing

The second byte of an instruction and the lower 8 bit of the index register are added in this mode. As for AIM, OIM, EIM and TIM, the third byte of an instruction and the lower 8 bits of the index register are added.

This carry is added to the upper 8 bit of the index register and the result is used for addressing the memory. The modified address is retained in the temporary address register, so the contents of the index register doesn't change. This is a 2-byte instruction except AIM, OIM, EIM and TIM (3-byte instruction).

Implied Addressing

An instruction itself specifies the address. This is, the instruction addresses a stack pointer, index register, etc. This is a one-byte instruction.

Relative Addressing

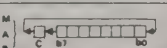
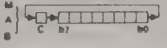
The second byte of an instruction and the lower 8 bits of the program counter are added. The carry or borrow is added to the upper 8 bit. So addressing from −126 to +129 byte of the current instruction is enabled. This is a 2-byte instruction.

(Note) CLI, SEI Instructions and Interrupt Operation
When accepting the IRQ at a preset timing with CLI and SEI instructions, more than 2 cycles are necessary between the CLI and SEI instructions. For example, the following program (a)(b) don't accept the IRQ but (c) accepts it.

.	.	.
.	.	.
.	.	.
.	.	CLI
CLI	CLI	NOP
SEI	NOP	NOP
.	SEI	SEI
.	.	.
.	.	.
.	.	.
.	.	.
(a)	(b)	(c)

The same thing can be said to the TAP instruction instead of the CLI and SEI instructions.

Table 11 Accumulator, Memory Manipulation Instructions

Operations	Mnemonic	Addressing Modes															Boolean/ Arithmetic Operation	Condition Code Register					
		IMMED			DIRECT			INDEX			EXTEND			IMPLIED				5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#		H	I	N	Z	V	C
Add	ADDA	8B	2	2	9B	3	2	AB	4	2	BB	4	3				A + M → A	↕	•	↕	↕	↕	↕
	ADDB	CB	2	2	DB	3	2	EB	4	2	FB	4	3				B + M → B	↕	•	↕	↕	↕	↕
Add Double	ADDD	C3	3	3	D3	4	2	E3	5	2	F3	5	3				A B + M M + 1 → A B	•	•	↕	↕	↕	↕
Add Accumulators	ABA													1B	1	1	A + B → A	↕	•	↕	↕	↕	↕
Add With Carry	ADCA	89	2	2	99	3	2	A9	4	2	B9	4	3				A + M + C → A	↕	•	↕	↕	↕	↕
	ADCB	C9	2	2	D9	3	2	E9	4	2	F9	4	3				B + M + C → B	↕	•	↕	↕	↕	↕
AND	ANDA	84	2	2	94	3	2	A4	4	2	B4	4	3				A·M → A	•	•	↕	↕	R	•
	ANDB	C4	2	2	D4	3	2	E4	4	2	F4	4	3				B·M → B	•	•	↕	↕	R	•
Bit Test	BIT A	85	2	2	95	3	2	A5	4	2	B5	4	3				A·M	•	•	↕	↕	R	•
	BIT B	C5	2	2	D5	3	2	E5	4	2	F5	4	3				B·M	•	•	↕	↕	R	•
Clear	CLR							6F	5	2	7F	5	3				00 → M	•	•	R	S	R	R
	CLRA													4F	1	1	00 → A	•	•	R	S	R	R
	CLRB													5F	1	1	00 → B	•	•	R	S	R	R
Compare	CMPA	81	2	2	91	3	2	A1	4	2	B1	4	3				A - M	•	•	↕	↕	↕	↕
	CMPB	C1	2	2	D1	3	2	E1	4	2	F1	4	3				B - M	•	•	↕	↕	↕	↕
Compare Accumulators	CBA													11	1	1	A - B	•	•	↕	↕	↕	↕
Complement, 1's	COM							63	6	2	73	6	3				$\overline{M}$ → M	•	•	↕	↕	R	S
	COMA													43	1	1	$\overline{A}$ → A	•	•	↕	↕	R	S
	COMB													53	1	1	$\overline{B}$ → B	•	•	↕	↕	R	S
Complement, 2's (Negate)	NEG							60	6	2	70	6	3				00 - M → M	•	•	↕	↕	①	②
	NEGA													40	1	1	00 - A → A	•	•	↕	↕	①	②
	NEGB													50	1	1	00 - B → B	•	•	↕	↕	①	②
Decimal Adjust, A	DAA													19	2	1	Converts binary add of BCD characters into BCD format	•	•	↕	↕	↕	③
Decrement	DEC							6A	6	2	7A	6	3				M - 1 → M	•	•	↕	↕	④	•
	DECA													4A	1	1	A - 1 → A	•	•	↕	↕	④	•
	DECB													5A	1	1	B - 1 → B	•	•	↕	↕	④	•
Exclusive OR	EORA	88	2	2	98	3	2	A8	4	2	B8	4	3				A ⊕ M → A	•	•	↕	↕	R	•
	EORB	C8	2	2	D8	3	2	E8	4	2	F8	4	3				B ⊕ M → B	•	•	↕	↕	R	•
Increment	INC							6C	6	2	7C	6	3				M + 1 → M	•	•	↕	↕	⑤	•
	INCA													4C	1	1	A + 1 → A	•	•	↕	↕	⑤	•
	INCB													5C	1	1	B + 1 → B	•	•	↕	↕	⑤	•
Load Accumulator	LDAA	86	2	2	96	3	2	A6	4	2	B6	4	3				M → A	•	•	↕	↕	R	•
	LDAB	C6	2	2	D6	3	2	E6	4	2	F6	4	3				M → B	•	•	↕	↕	R	•
Load Double Accumulator	LDD	CC	3	3	DC	4	2	EC	5	2	FC	5	3				M + 1 → B, M → A	•	•	↕	↕	R	•
Multiply Unsigned	MUL													3D	7	1	A x B → A B	•	•	•	•	•	⑪
OR, Inclusive	ORAA	8A	2	2	9A	3	2	AA	4	2	BA	4	3				A + M → A	•	•	↕	↕	R	•
	ORAB	CA	2	2	DA	3	2	EA	4	2	FA	4	3				B + M → B	•	•	↕	↕	R	•
Push Data	PSHA													36	4	1	A → Msp, SP - 1 → SP	•	•	•	•	•	•
	PSHB													37	4	1	B → Msp, SP - 1 → SP	•	•	•	•	•	•
Pull Data	PULA													32	3	1	SP + 1 → SP, Msp → A	•	•	•	•	•	•
	PULB													33	3	1	SP + 1 → SP, Msp → B	•	•	•	•	•	•
Rotate Left	ROL							69	6	2	79	6	3					•	•	↕	↕	⑥	↕
	ROLA													49	1	1		•	•	↕	↕	⑥	↕
	ROLB													59	1	1		•	•	↕	↕	⑥	↕
Rotate Right	ROR							66	6	2	76	6	3					•	•	↕	↕	⑦	↕
	RORA													46	1	1		•	•	↕	↕	⑦	↕
	RORB													56	1	1		•	•	↕	↕	⑦	↕

(Note) Condition Code Register will be explained in Note of Table 14.

(continued)

Table 11 Accumulator, Memory Manipulation Instructions

Operations	Mnemonic	IMMED OP	IMMED ~	IMMED #	DIRECT OP	DIRECT ~	DIRECT #	INDEX OP	INDEX ~	INDEX #	EXTEND OP	EXTEND ~	EXTEND #	IMPLIED OP	IMPLIED ~	IMPLIED #	Boolean/ Arithmetic Operation	5 H	4 I	3 N	2 Z	1 V	0 C
Shift Left Arithmetic	ASL							68	6	2	78	6	3				M	•	•	↕	↕	⑤	↕
	ASLA													48	1	1	A	•	•	↕	↕	⑤	↕
	ASLB													58	1	1	B	•	•	↕	↕	⑤	↕
Double Shift Left, Arithmetic	ASLD													05	1	1	ACC A/ ACC B	•	•	↕	↕	⑥	↕
Shift Right Arithmetic	ASR							67	6	2	77	6	3				M	•	•	↕	↕	⑤	↕
	ASRA													47	1	1	A	•	•	↕	↕	⑤	↕
	ASRB													57	1	1	B	•	•	↕	↕	⑤	↕
Shift Right Logical	LSR							64	6	2	74	6	3				M	•	•	R	↕	⑤	↕
	LSRA													44	1	1	A	•	•	R	↕	⑤	↕
	LSRB													54	1	1	B	•	•	R	↕	⑤	↕
Double Shift Right Logical	LSRD													04	1	1	ACC A/ ACC B	•	•	R	↕	⑥	↕
Store Accumulator	STAA				97	3	2	A7	4	2	B7	4	3				A → M	•	•	↕	↕	R	•
	STAB				D7	3	2	E7	4	2	F7	4	3				B → M	•	•	↕	↕	R	•
Store Double Accumulator	STD				DD	4	2	ED	5	2	FD	5	3				A → M B → M + 1	•	•	↕	↕	R	•
Subtract	SUBA	80	2	2	90	3	2	A0	4	2	B0	4	3				A − M → A	•	•	↕	↕	↕	↕
	SUBB	C0	2	2	D0	3	2	E0	4	2	F0	4	3				B − M → B	•	•	↕	↕	↕	↕
Double Subtract	SUBD	83	3	3	93	4	2	A3	5	2	B3	5	3				A B − M M + 1 → A B	•	•	↕	↕	↕	↕
Subtract Accumulators	SBA													10	1	1	A − B → A	•	•	↕	↕	↕	↕
Subtract With Carry	SBCA	82	2	2	92	3	2	A2	4	2	B2	4	3				A − M − C → A	•	•	↕	↕	↕	↕
	SBCB	C2	2	2	D2	3	2	E2	4	2	F2	4	3				B − M − C → B	•	•	↕	↕	↕	↕
Transfer Accumulators	TAB													16	1	1	A → B	•	•	↕	↕	R	•
	TBA													17	1	1	B → A	•	•	↕	↕	R	•
Test Zero or Minus	TST							6D	4	2	7D	4	3				M − 00	•	•	↕	↕	R	R
	TSTA													4D	1	1	A − 00	•	•	↕	↕	R	R
	TSTB													5D	1	1	B − 00	•	•	↕	↕	R	R
And Immediate	AIM				71	6	3	61	7	3							M·IMM→M	•	•	↕	↕	R	•
OR Immediate	OIM				72	6	3	62	7	3							M+IMM→M	•	•	↕	↕	R	•
EOR Immediate	EIM				75	6	3	65	7	3							M⊕IMM→M	•	•	↕	↕	R	•
Test Immediate	TIM				7B	4	3	6B	5	3							M·IMM	•	•	↕	↕	R	•

(Note) Condition Code Register will be explained in Note of Table 14.

- **Additional Instruction**

In addition to the HD6801 instruction set, the HD6303Y prepares the following new instructions.

AIM ..... (M)·(IMM) → (M)

Executes "AND" operation to immediate data and the memory contents and stores its result in the memory.

OIM ..... (M)+(IMM) → (M)

Executes "OR" operation to immediate data and the memory contents and stores its result in the memory.

EIM ..... (M) ⊕ (IMM) → (M)

Executes "EOR" operation to immediate data and the memory contents and stores its result in the memory.

TIM ..... (M)·(IMM)

Executes "AND" operation to immediate data and changes the relative flag of the condition code register.

These are the 3-byte instructions; the first byte is op code, the second immediate data and the third address modifier.

XGDX ..... (ACCD)↔(IX)

Exchanges the contents of accumulator and the index register.

SLP

Goes to the sleep mode. Refer to "LOW POWER DISSIPATION MODE" for more details of the sleep mode.

Table 12 Index Register, Stack Manipulation Instructions

Pointer Operations	Mnemonic	Addressing Modes															Boolean/ Arithmetic Operation	Condition Code Register					
		IMMED			DIRECT			INDEX			EXTEND			IMPLIED				5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#		H	I	N	Z	V	C
Compare Index Reg	CPX	8C	3	3	9C	4	2	AC	5	2	BC	5	3				X − M M+1	•	•	↕	↕	↕	↕
Decrement Index Reg	DEX													09	1	1	X − 1 → X	•	•	•	↕	•	•
Decrement Stack Pntr	DES													34	1	1	SP − 1 → SP	•	•	•	•	•	•
Increment Index Reg	INX													08	1	1	X + 1 → X	•	•	•	↕	•	•
Increment Stack Pntr	INS													31	1	1	SP + 1 → SP	•	•	•	•	•	•
Load Index Reg	LDX	CE	3	3	DE	4	2	EE	5	2	FE	5	3				M → X_H, (M + 1) → X_L	•	•	⑦	↕	R	•
Load Stack Pntr	LDS	8E	3	3	9E	4	2	AE	5	2	BE	5	3				M → SP_H, (M+1) → SP_L	•	•	⑦	↕	R	•
Store Index Reg	STX				DF	4	2	EF	5	2	FF	5	3				X_H → M, X_L → (M + 1)	•	•	⑦	↕	R	•
Store Stack Pntr	STS				9F	4	2	AF	5	2	BF	5	3				SP_H → M, SP_L → (M + 1)	•	•	⑦	↕	R	•
Index Reg → Stack Pntr	TXS													35	1	1	X − 1 → SP	•	•	•	•	•	•
Stack Pntr → Index Reg	TSX													30	1	1	SP + 1 → X	•	•	•	•	•	•
Add	ABX													3A	1	1	B + X → X	•	•	•	•	•	•
Push Data	PSHX													3C	5	1	X_L → M_{SP}, SP − 1 → SP X_H → M_{SP}, SP − 1 → SP	•	•	•	•	•	•
Pull Data	PULX													38	4	1	SP + 1 → SP, M_{SP} → X_H SP + 1 → SP, M_{SP} → X_L	•	•	•	•	•	•
Exchange	XGDX													18	2	1	ACCD·↔IX	•	•	•	•	•	•

(Note) Condition Code Register will be explained in Note of Table 14.

Table 13 Jump, Branch Instruction

Operations	Mnemonic	Addressing Modes															Branch Test	Condition Code Register					
		RELATIVE			DIRECT			INDEX			EXTEND			IMPLIED				5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#		H	I	N	Z	V	C
Branch Always	BRA	20	3	2													None	•	•	•	•	•	•
Branch Never	BRN	21	3	2													None	•	•	•	•	•	•
Branch If Carry Clear	BCC	24	3	2													C = 0	•	•	•	•	•	•
Branch If Carry Set	BCS	25	3	2													C = 1	•	•	•	•	•	•
Branch If = Zero	BEQ	27	3	2													Z = 1	•	•	•	•	•	•
Branch If ≥ Zero	BGE	2C	3	2													N ⊕ V = 0	•	•	•	•	•	•
Branch If > Zero	BGT	2E	3	2													Z + (N ⊕ V) = 0	•	•	•	•	•	•
Branch If Higher	BHI	22	3	2													C + Z = 0	•	•	•	•	•	•
Branch If ≤ Zero	BLE	2F	3	2													Z + (N ⊕ V) = 1	•	•	•	•	•	•
Branch If Lower Or Same	BLS	23	3	2													C + Z = 1	•	•	•	•	•	•
Branch If < Zero	BLT	2D	3	2													N ⊕ V = 1	•	•	•	•	•	•
Branch If Minus	BMI	2B	3	2													N = 1	•	•	•	•	•	•
Branch If Not Equal Zero	BNE	26	3	2													Z = 0	•	•	•	•	•	•
Branch If Overflow Clear	BVC	28	3	2													V = 0	•	•	•	•	•	•
Branch If Overflow Set	BVS	29	3	2													V = 1	•	•	•	•	•	•
Branch If Plus	BPL	2A	3	2													N = 0	•	•	•	•	•	•
Branch To Subroutine	BSR	8D	5	2														•	•	•	•	•	•
Jump	JMP							6E	3	2	7E	3	3					•	•	•	•	•	•
Jump To Subroutine	JSR				9D	5	2	AD	5	2	BD	6	3					•	•	•	•	•	•
No Operation	NOP													01	1	1	Advances Prog. Cntr. Only	•	•	•	•	•	•
Return From Interrupt	RTI													3B	10	1		——— ⑧ ———					
Return From Subroutine	RTS													39	5	1		•	•	•	•	•	•
Software Interrupt	SWI													3F	12	1		•	S	•	•	•	•
Wait for Interrupt*	WAI													3E	9	1		•	⑨	•	•	•	•
Sleep	SLP													1A	4	1		•	•	•	•	•	•

(Note) * WAI puts R/$\overline{W}$ high; Address Bus goes to FFFF; Data Bus goes to the three state.
Condition Code Register will be explained in Note of Table 14.

Table 14 Condition Code Register Manipulation Instructions

Operations	Mnemonic	Addressing Modes IMPLIED OP	~	#	Boolean Operation	Condition Code Register 5 H	4 I	3 N	2 Z	1 V	0 C
Clear Carry	CLC	0C	1	1	0 → C	•	•	•	•	•	R
Clear Interrupt Mask	CLI	0E	1	1	0 → I	•	R	•	•	•	•
Clear Overflow	CLV	0A	1	1	0 → V	•	•	•	•	R	•
Set Carry	SEC	0D	1	1	1 → C	•	•	•	•	•	S
Set Interrupt Mask	SEI	0F	1	1	1 → I	•	S	•	•	•	•
Set Overflow	SEV	0B	1	1	1 → V	•	•	•	•	S	•
Accumulator A → CCR	TAP	06	1	1	A → CCR	——— ⑧ ———					
CCR → Accumulator A	TPA	07	1	1	CCR → A	•	•	•	•	•	•

LEGEND

OP Operation Code (Hexadecimal)
~ Number of MCU Cycles
M_{SP} Contents of memory location pointed by Stack Pointer
Number of Program Bytes
+ Arithmetic Plus
– Arithmetic Minus
• Boolean AND
+ Boolean Inclusive OR
⊕ Boolean Exclusive OR
$\overline{M}$ Complement of M
→ Transfer into
0 Bit = Zero
00 Byte = Zero

CONDITION CODE SYMBOLS

H Half-carry from bit 3 to bit 4
I Interrupt mask
N Negative (sign bit)
Z Zero (byte)
V Overflow, 2's complement
C Carry/Borrow from/to bit 7
R Reset Always
S Set Always
↕ Set if true after test or clear
• Not Affected

(Note) Condition Code Register Notes: (Bit set if test is true and cleared otherwise)

① (Bit V) Test: Result = 10000000?
② (Bit C) Test: Result ≠ 00000000?
③ (Bit C) Test: BCD Character of high-order byte greater than 10? (Not cleared if previously set)
④ (Bit V) Test: Operand = 10000000 prior to execution?
⑤ (Bit V) Test: Operand = 01111111 prior to execution?
⑥ (Bit V) Test: Set equal to N⊕C = 1 after the execution of instructions
⑦ (Bit N) Test: Result less than zero? (Bit 15=1)
⑧ (All Bit) Load Condition Code Register from Stack.
⑨ (Bit I) Set when interrupt occurs. If previously set, a Non-Maskable Interrupt is required to exit the wait state.
⑩ (All Bit) Set according to the contents of Accumulator A.
⑪ (Bit C) Result of Multiplication Bit 7=1? (ACCB)

Table 15 OP-Code Map

OP CODE HI	LO	0000 0	0001 1	0010 2	0011 3	ACC A 0100 4	ACC B 0101 5	IND 0110 6	EXT/DIR 0111 7	ACCA or SP IMM 1000 8	DIR 1001 9	IND A	EXT B	ACCB or X IMM 1100 C	DIR 1101 D	IND E	EXT F	HI
0000	0		SBA	BRA	TSX	NEG				SUB								0
0001	1	NOP	CBA	BRN	INS			AIM		CMP								1
0010	2			BHI	PULA			OIM		SBC								2
0011	3			BLS	PULB	COM				SUBD				ADDD				3
0100	4	LSRD		BCC	DES	LSR				AND								4
0101	5	ASLD		BCS	TXS			EIM		BIT								5
0110	6	TAP	TAB	BNE	PSHA	ROR				LDA								6
0111	7	TPA	TBA	BEQ	PSHB	ASR				STA				STA				7
1000	8	INX	XGDX	BVC	PULX	ASL				EOR								8
1001	9	DEX	DAA	BVS	RTS	ROL				ADC								9
1010	A	CLV	SLP	BPL	ABX	DEC				ORA								A
1011	B	SEV	ABA	BMI	RTI			TIM		ADD								B
1100	C	CLC		BGE	PSHX	INC				CPX				LDD				C
1101	D	SEC		BLT	MUL	TST				BSR	JSR			STD				D
1110	E	CLI		BGT	WAI			JMP		LDS				LDX				E
1111	F	SEI		BLE	SWI	CLR				STS				STX				F
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	

UNDEFINED OP CODE

* Only each instructions of AIM, OIM, EIM, TIM

■ CPU OPERATION

• CPU Instruction Flow

When operating, the CPU fetches an instrution from a memory and executes the required function. This sequence starts with $\overline{RES}$ cancel and repeats itself limitlessly if not affected by a special instruction or a control signal. SWI, RTI, WAI and SLP instructions change this operation, while $\overline{NMI}$, $\overline{IRQ_1}$, $\overline{IRQ_2}$, IRQ_3, $\overline{HALT}$ and $\overline{STBY}$ control it. Fig. 29 gives the CPU mode transition and Fig. 30 the CPU system flow chart. Table 16 shows CPU operating states and port states.

• Operation at Each Instruction Cycle

Table 17 shows the operation at each instruction cycle. By the pipeline control of the HD6303Y, MULT, PUL, DAA and XGDX instructions, etc. prefetch the next instruction. So attention is necessary to the counting of the instruction cycles because it is different from the usual one—from op code fetch to the next instruction op code.

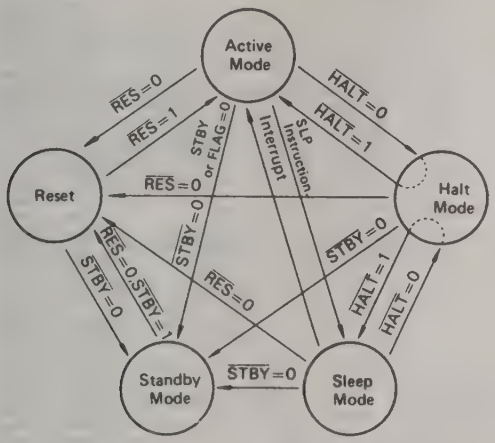
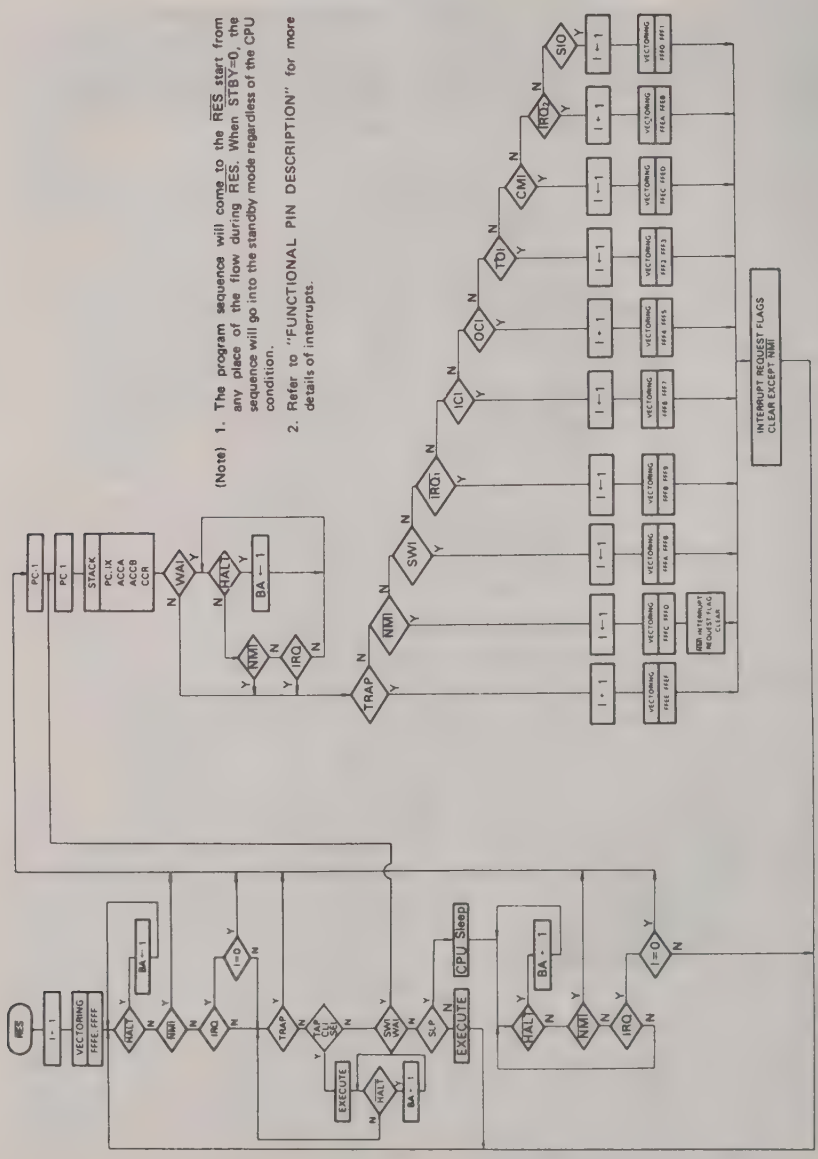


Figure 29 CPU Operation Mode Transition

Table 16 CPU Operation State and Port, Bus, Control Signal State

Port	Reset	STBY*3	HALT	Sleep
$A_0 \sim A_7$	H	T	T	H
Port 2	T	T	Keep	Keep
$D_0 \sim D_7$	T	T	T	T
$A_8 \sim A_{15}$	H	T	T	H
Port 5	T	T	Keep	Keep
Port 6	T	T	Keep	Keep
Control Signal	*1	T	*2	*1

*1 $\overline{RD}$, $\overline{WR}$, R/$\overline{W}$, $\overline{DIR}$ = H, BA = L
*2 $\overline{RD}$, $\overline{WR}$, R/$\overline{W}$ = T, $\overline{DIR}$, BA = H
*3 E pin goes to high impedance state.



(Note) 1. The program sequence will come to the $\overline{RES}$ start from any place of the flow during $\overline{RES}$. When STBY=0, the sequence will go into the standby mode regardless of the CPU condition.

2. Refer to "FUNCTIONAL PIN DESCRIPTION" for more details of interrupts.

Figure 30 HD6303Y System Flow Chart

Table 17 Cycle-by-Cycle Operation

Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/$\overline{W}$	$\overline{RD}$	$\overline{WR}$	$\overline{LIR}$	Data Bus
IMMEDIATE								
ADC ADD AND BIT CMP EOR LDA ORA SBC SUB	2	1	Op Code Address+1	1	0	1	1	Operand Data
		2	Op Code Address+2	1	0	1	0	Next Op Code
ADDD CPX LDD LDS LDX SUBD	3	1	Op Code Address+1	1	0	1	1	Operand Data (MSB)
		2	Op Code Address+2	1	0	1	1	Operand Data (LSB)
		3	Op Code Address+3	1	0	1	0	Next Op Code
DIRECT								
ADC ADD AND BIT CMP EOR LDA ORA SBC SUB	3	1	Op Code Address+1	1	0	1	1	Address of Operand (LSB)
		2	Address of Operand	1	0	1	1	Operand Data
		3	Op Code Address+2	1	0	1	0	Next Op Code
STA	3	1	Op Code Address+1	1	0	1	1	Destination Address
		2	Destination Address	0	1	0	1	Accumulator Data
		3	Op Code Address+2	1	0	1	0	Next Op Code
ADDD CPX LDD LDS LDX SUBD	4	1	Op Code Address+1	1	0	1	1	Address of Operand (LSB)
		2	Address of Operand	1	0	1	1	Operand Data (MSB)
		3	Address of Operand+1	1	0	1	1	Operand Data (LSB)
		4	Op Code Address+2	1	0	1	0	Next Op Code
STD STS STX	4	1	Op Code Address+1	1	0	1	1	Destination Address (LSB)
		2	Destination Address	0	1	0	1	Register Data (MSB)
		3	Destination Address+1	0	1	0	1	Register Data (LSB)
		4	Op Code Address+2	1	0	1	0	Next Op Code
JSR	5	1	Op Code Address+1	1	0	1	1	Jump Address (LSB)
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	Stack Pointer	0	1	0	1	Return Address (LSB)
		4	Stack Pointer−1	0	1	0	1	Return Address (MSB)
		5	Jump Address	1	0	1	0	First Subroutine Op Code
TIM	4	1	Op Code Address+1	1	0	1	1	Immediate Data
		2	Op Code Address+2	1	0	1	1	Address of Operand (LSB)
		3	Address of Operand	1	0	1	1	Operand Data
		4	Op Code Address+3	1	0	1	0	Next Op Code
AIM EIM OIM	6	1	Op Code Address+1	1	0	1	1	Immediate Data
		2	Op Code Address+2	1	0	1	1	Address of Operand (LSB)
		3	Address of Operand	1	0	1	1	Operand Data
		4	FFFF	1	1	1	1	Restart Address (LSB)
		5	Address of Operand	0	1	0	1	New Operand Data
		6	Op Code Address+3	1	0	1	0	Next Op Code

(Continued)

Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/$\overline{W}$	$\overline{RD}$	$\overline{WR}$	$\overline{LIR}$	Data Bus
INDEXED								
JMP	3	1	Op Code Address+1	1	0	1	1	Offset
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	Jump Address	1	0	1	0	First Op Code of Jump Routine
ADC ADD AND BIT CMP EOR LDA ORA SBC SUB TST	4	1	Op Code Address+1	1	0	1	1	Offset
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	IX+Offset	1	0	1	1	Operand Data
		4	Op Code Address+2	1	0	1	0	Next Op Code
STA	4	1	Op Code Address+1	1	0	1	1	Offset
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	IX+Offset	0	1	0	1	Accumulator Data
		4	Op Code Address+2	1	0	1	0	Next Op Code
ADDD CPX LDD LDS LDX SUBD	5	1	Op Code Address+1	1	0	1	1	Offset
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	IX+Offset	1	0	1	1	Operand Data (MSB)
		4	IX+Offset+1	1	0	1	1	Operand Data (LSB)
		5	Op Code Address+2	1	0	1	0	Next Op Code
STD STS STX	5	1	Op Code Address+1	1	0	1	1	Offset
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	IX+Offset	0	1	0	1	Register Data (MSB)
		4	IX+Offset+1	0	1	0	1	Register Data (LSB)
		5	Op Code Address+2	1	0	1	0	Next Op Code
JSR	5	1	Op Code Address+1	1	0	1	1	Offset
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	Stack Pointer	0	1	0	1	Return Address (LSB)
		4	Stack Pointer−1	0	1	0	1	Return Address (MSB)
		5	IX+Offset	1	0	1	0	First Subroutine Op Code
ASL ASR COM DEC INC LSR NEG ROL ROR	6	1	Op Code Address+1	1	0	1	1	Offset
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	IX+Offset	1	0	1	1	Operand Data
		4	FFFF	1	1	1	1	Restart Address (LSB)
		5	IX+Offset	0	1	0	1	New Operand Data
		6	OP Code Address+2	1	0	1	0	Next Op Code
TIM	5	1	Op Code Address+1	1	0	1	1	Immediate Data
		2	Op Code Address+2	1	0	1	1	Offset
		3	FFFF	1	1	1	1	Restart Address (LSB)
		4	IX+Offset	1	0	1	1	Operand Data
		5	Op Code Address+3	1	0	1	0	Next Op Code
CLR	5	1	Op Code Address+1	1	0	1	1	Offset
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	IX+Offset	1	0	1	1	Operand Data
		4	IX+Offset	0	1	0	1	00
		5	Op Code Address+2	1	0	1	0	Next Op Code
AIM EIM OIM	7	1	Op Code Address+1	1	0	1	1	Immediate Data
		2	Op Code Address+2	1	0	1	1	Offset
		3	FFFF	1	1	1	1	Restart Address (LSB)
		4	IX+Offset	1	0	1	1	Operand Data
		5	FFFF	1	1	1	1	Restart Address (LSB)
		6	IX+Offset	0	1	0	1	New Operand Data
		7	Op Code Address+3	1	0	1	0	Next Op Code

(Continued)

Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/$\overline{W}$	$\overline{RD}$	$\overline{WR}$	$\overline{LIR}$	Data Bus
EXTEND								
JMP	3	1	Op Code Address+1	1	0	1	1	Jump Address (MSB)
		2	Op Code Address+2	1	0	1	1	Jump Address (LSB)
		3	Jump Address	1	0	1	0	Next Op Code
ADC ADD TST AND BIT CMP EOR LDA ORA SBC SUB	4	1	Op Code Address+1	1	0	1	1	Address of Operand (MSB)
		2	Op Code Address+2	1	0	1	1	Address of Operand (LSB)
		3	Address of Operand	1	0	1	1	Operand Data
		4	Op Code Address+3	1	0	1	0	Next Op Code
STA	4	1	Op Code Address+1	1	0	1	1	Destination Address (MSB)
		2	Op Code Address+2	1	0	1	1	Destination Address (LSB)
		3	Destination Address	0	1	0	1	Accumulator Data
		4	Op Code Address+3	1	0	1	0	Next Op Code
ADDD CPX LDD LDS LDX SUBD	5	1	Op Code Address+1	1	0	1	1	Address of Operand (MSB)
		2	Op Code Address+2	1	0	1	1	Address of Operand (LSB)
		3	Address of Operand	1	0	1	1	Operand Data (MSB)
		4	Address of Operand+1	1	0	1	1	Operand Data (LSB)
		5	Op Code Address+3	1	0	1	0	Next Op Code
STD STS STX	5	1	Op Code Address+1	1	0	1	1	Destination Address (MSB)
		2	Op Code Address+2	1	0	1	1	Destination Address (LSB)
		3	Destination Address	0	1	0	1	Register Data (MSB)
		4	Destination Address+1	0	1	0	1	Register Data (LSB)
		5	Op Code Address+3	1	0	1	0	Next Op Code
JSR	6	1	Op Code Address+1	1	0	1	1	Jump Address (MSB)
		2	Op Code Address+2	1	0	1	1	Jump Address (LSB)
		3	FFFF	1	1	1	1	Restart Address (LSB)
		4	Stack Pointer	0	1	0	1	Return Address (LSB)
		5	Stack Pointer−1	0	1	0	1	Return Address (MSB)
		6	Jump Address	1	0	1	0	First Subroutine Op Code
ASL ASR COM DEC INC LSR NEG ROL ROR	6	1	Op Code Address+1	1	0	1	1	Address of Operand (MSB)
		2	Op Code Address+2	1	0	1	1	Address of Operand (LSB)
		3	Address of Operand	1	0	1	1	Operand Data
		4	FFFF	1	1	1	1	Restart Address (LSB)
		5	Address of Operand	0	1	0	1	New Operand Data
		6	Op Code Address+3	1	0	1	0	Next Op Code
CLR	5	1	Op Code Address+1	1	0	1	1	Address of Operand (MSB)
		2	Op Code Address+2	1	0	1	1	Address of Operand (LSB)
		3	Address of Operand	1	0	1	1	Operand Data
		4	Address of Operand	0	1	0	1	00
		5	Op Code Address+3	1	0	1	0	Next Op Code

(Continued)

Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/$\overline{W}$	$\overline{RD}$	$\overline{WR}$	$\overline{LIR}$	Data Bus
IMPLIED								
ABA ABX ASL ASLD ASR CBA CLC CLI CLR CLV COM DEC DES DEX INC INS INX LSR LSRD ROL ROR NOP SBA SEC SEI SEV TAB TAP TBA TPA TST TSX TXS	1	1	Op Code Address+1	1	0	1	0	Next Op Code
DAA XGDX	2	1	Op Code Address+1	1	0	1	0	Next Op Code
		2	FFFF	1	1	1	1	Restart Address (LSB)
PULA PULB	3	1	Op Code Address+1	1	0	1	0	Next Op Code
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	Stack Pointer+1	1	0	1	1	Data from Stack
PSHA PSHB	4	1	Op Code Address+1	1	0	1	1	Next Op Code
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	Stack Pointer	0	1	0	1	Accumulator Data
		4	Op Code Address+1	1	0	1	0	Next Op Code
PULX	4	1	Op Code Address+1	1	0	1	0	Next Op Code
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	Stack Pointer+1	1	0	1	1	Data from Stack (MSB)
		4	Stack Pointer+2	1	0	1	1	Data from Stack (LSB)
PSHX	5	1	Op Code Address+1	1	0	1	1	Next Op Code
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	Stack Pointer	0	1	0	1	Index Register (LSB)
		4	Stack Pointer−1	0	1	0	1	Index Register (MSB)
		5	Op Code Address+1	1	0	1	0	Next Op Code
RTS	5	1	Op Code Address+1	1	0	1	1	Next Op Code
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	Stack Pointer+1	1	0	1	1	Return Address (MSB)
		4	Stack Pointer+2	1	0	1	1	Return Address (LSB)
		5	Return Address	1	0	1	0	First Op Code of Return Routine
MUL	7	1	Op Code Address+1	1	0	1	0	Next Op Code
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	FFFF	1	1	1	1	Restart Address (LSB)
		4	FFFF	1	1	1	1	Restart Address (LSB)
		5	FFFF	1	1	1	1	Restart Address (LSB)
		6	FFFF	1	1	1	1	Restart Address (LSB)
		7	FFFF	1	1	1	1	Restart Address (LSB)

(Continued)

Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/$\overline{W}$	$\overline{RD}$	$\overline{WR}$	$\overline{LIR}$	Data Bus
IMPLIED								
WAI	9	1	Op Code Address+1	1	0	1	1	Next Op Code
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	Stack Pointer	0	1	0	1	Return Address (LSB)
		4	Stack Pointer−1	0	1	0	1	Return Address (MSB)
		5	Stack Pointer−2	0	1	0	1	Index Register (LSB)
		6	Stack Pointer−3	0	1	0	1	Index Register (MSB)
		7	Stack Pointer−4	0	1	0	1	Accumulator A
		8	Stack Pointer−5	0	1	0	1	Accumulator B
		9	Stack Pointer−6	0	1	0	1	Conditional Code Register
RTI	10	1	Op Code Address+1	1	0	1	1	Next Op Code
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	Stack Pointer+1	1	0	1	1	Conditional Code Register
		4	Stack Pointer+2	1	0	1	1	Accumulator B
		5	Stack Pointer+3	1	0	1	1	Accumulator A
		6	Stack Pointer+4	1	0	1	1	Index Register (MSB)
		7	Stack Pointer+5	1	0	1	1	Index Register (LSB)
		8	Stack Pointer+6	1	0	1	1	Return Address (MSB)
		9	Stack Pointer+7	1	0	1	1	Return Address (LSB)
		10	Return Address	1	0	1	0	First Op Code of Return Routine
SWI	12	1	Op Code Address+1	1	0	1	1	Next Op Code
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	Stack Pointer	0	1	0	1	Return Address (LSB)
		4	Stack Pointer−1	0	1	0	1	Return Address (MSB)
		5	Stack Pointer−2	0	1	0	1	Index Register (LSB)
		6	Stack Pointer−3	0	1	0	1	Index Register (MSB)
		7	Stack Pointer−4	0	1	0	1	Accumulator A
		8	Stack Pointer−5	0	1	0	1	Accumulator B
		9	Stack Pointer−6	0	1	0	1	Conditional Code Register
		10	Vector Address FFFA	1	0	1	1	Address of SWI Routine (MSB)
		11	Vector Address FFFB	1	0	1	1	Address of SWI Routine (LSB)
		12	Address of SWI Routine	1	0	1	0	First Op Code of SWI Routine
SLP	4	1	Op Code Address+1	1	0	1	1	Next Op Code
		2	FFFF	1	1	1	1	Restart Address (LSB)
		↑ Sleep ↓						
		3	FFFF	1	1	1	1	Restart Address (LSB)
		4	Op Code Address+1	1	0	1	0	Next Op Code
RELATIVE								
BCC BCS BEQ BGE BGT BHI BLE BLS BLT BMT BNE BPL BRA BRN BVC BVS	3	1	Op Code Address+1	1	0	1	1	Branch Offset
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	Branch Address ····Test="1" Op Code Address+1 ··Test="0"	1	0	1	0	First Op Code of Branch Routine Next Op Code
BSR	5	1	Op Code Address+1	1	0	1	1	Offset
		2	FFFF	1	1	1	1	Restart Address (LSB)
		3	Stack Pointer	0	1	0	1	Return Address (LSB)
		4	Stack Pointer−1	0	1	0	1	Return Address (MSB)
		5	Branch Address	1	0	1	0	First Op Code of Subroutine

■ PRECAUTION TO THE BOARD DESIGN OF OSCILLATION CIRCUIT

As shown in Fig. 31, there is a case that the cross talk disturbs the normal oscillation if signal lines are put near the oscillation circuit. When designing a board, pay attention to this. Crystal and C_L must be put as near the HD6303Y as possible.

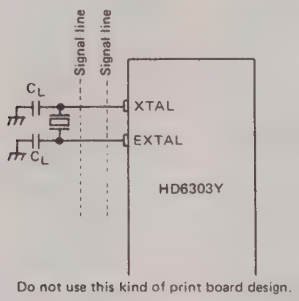


Figure 31 Precaution to the boad design of oscillation circuit

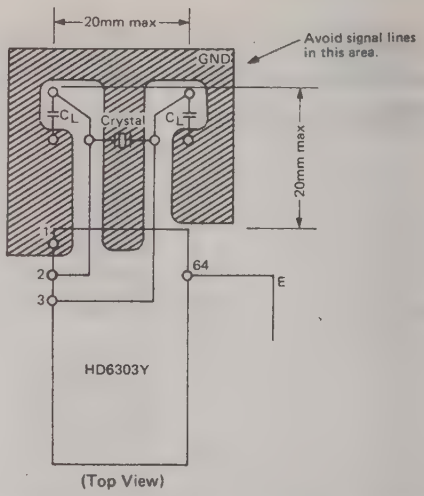


Figure 32 Example of Oscillation Circuits in Board Design

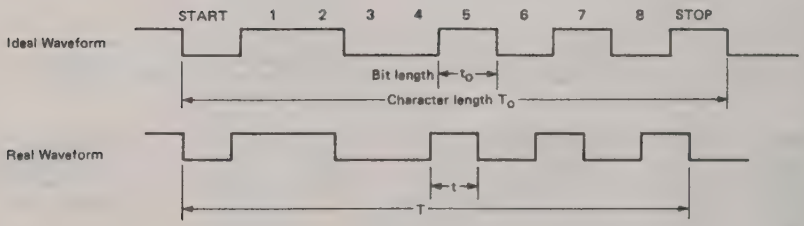
■ RECEIVE MARGIN OF THE SCI

Receive margin of the SCI contained in the HD6303Y is shown in Table 18.

Note: SCI = Serial Communication Interface

Table 18

	Bit distortion tolerance (t−to) /to	Character distortion tolerance (T−To) /To
HD6303Y	±43.7%	±4.37%



HD6305X2, HD63A05X2, HD63B05X2
CMOS MPU (Micro Processing Unit)

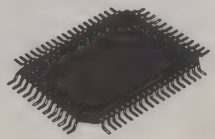
The HD6305X2 is a CMOS 8-bit micro processing unit. A CPU, a clock generator, a 128-byte RAM, I/O terminals, two timers and a serial communication interface (SCI) are built in the HD6305X2.

The HD6305X2 has the same functions as the HD6305X0's except for the number of I/O terminals. The HD6305X2 is a micro processing unit and its memory space is expandable to 16k bytes externally.

HD6305X2P, HD63A05X2P, HD63B05X2P

(DP-64S)

HD6305X2F, HD63A05X2F, HD63B05X2F



(FP-64)

■ HARDWARE FEATURES

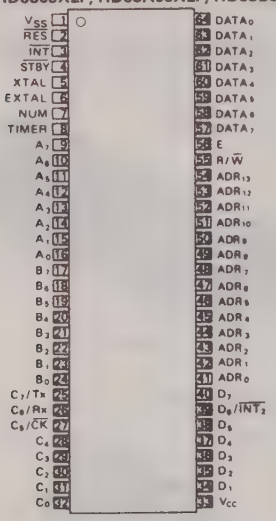
- 8-bit based MPU
- 128-bytes of RAM
- A total of 31 terminals, including 24 I/O's, 7 inputs
- Two timers
  - 8-bit timer with a 7-bit prescaler (programmable prescaler; event counter)
  - 15-bit timer (commonly used with the SCI clock divider)
- On-chip serial interface circuit (synchronized with clock)
- Six interrupts (two external, two timer, one serial and one software)
- Low power dissipation modes
  - Wait . . . . In this mode, the clock oscillator is on and the CPU halts but the timer/serial/interrupt function is operatable.
  - Stop . . . . In this mode, the clock stops but the RAM data, I/O status and registers are held.
  - Standby . . In this mode, the clock stops, the RAM data is held, and the other internal condition is reset.
- Minimum instruction cycle time
  - HD6305X2 . . . . . 1 μs (f = 1 MHz)
  - HD63A05X2 . . . . 0.67 μs (f = 1.5 MHz)
  - HD63B05X2 . . . . 0.5 μs (f = 2 MHz)
- Wide operating range
  V_{CC} = 3 to 6V (f = 0.1 to 0.5 MHz)
  - HD6305X2 . . . . . f = 0.1 to 1 MHz (V_{CC} = 5V ± 10%)
  - HD63A05X2 . . . . . f = 0.1 to 1.5 MHz (V_{CC} = 5V ± 10%)
  - HD63B05X2 . . . . . f = 0.1 to 2 MHz (V_{CC} = 5V ± 10%)
- System development fully supported by an evaluation kit

■ SOFTWARE FEATURES

- Similar to HD6800
- Byte efficient instruction set
- Powerful bit manipulation instructions (Bit Set, Bit Clear, and Bit Test and Branch usable for all RAM bits and all I/O terminals)
- A variety of interrupt operations
- Index addressing mode useful for table processing
- A variety of conditional branch instructions
- Ten powerful addressing modes
- All addressing modes adaptable to RAM, and I/O instructions
- Three new instructions, STOP, WAIT and DAA, added to the HD6805 family instruction set
- Instructions that are upward compatible with those of Motorola's MC6805P2 and MC146805G2

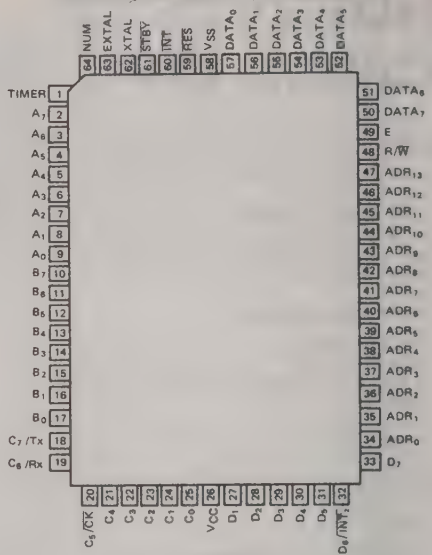
■ PIN ARRANGEMENT

● HD6305X2P, HD63A05X2P, HD63B05X2P



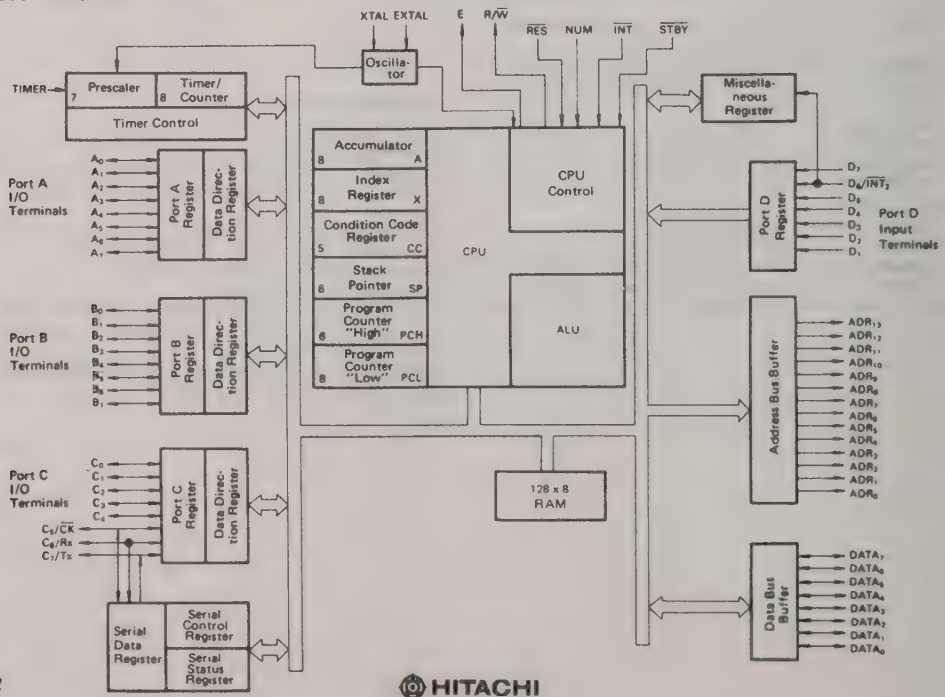
(Top View)

● HD6305X2F, HD63A05X2F, HD63B05X2F



(Top View)

■ BLOCK DIAGRAM



■ ABSOLUTE MAXIMUM RATINGS

Item	Symbol	Value	Unit
Supply Voltage	V_{CC}	−0.3 ~ +7.0	V
Input Voltage	V_{in}	−0.3 ~ V_{CC} + 0.3	V
Operating Temperature	T_{opr}	0 ~ +70	°C
Storage Temperature	T_{stg}	−55 ~ +150	°C

[NOTE] These products have a protection circuit in their input terminals against high electrostatic voltage or high electric fields. Notwithstanding, be careful not to apply any voltage higher than the absolute maximum rating to these high input impedance circuits. To assure normal operation, we recommended V_{in}, V_{out}; $V_{SS} \leq (V_{in}$ or $V_{out}) \leq V_{CC}$.

■ ELECTRICAL CHARACTERISTICS

● DC CHARACTERISTICS (V_{CC} = 5.0V±10%, V_{SS} = 0V, Ta = 0 ~ +70°C, unless otherwise noted.)

Item		Symbol	Test Condition	min	typ	max	Unit
Input "High" Voltage	$\overline{RES}$, $\overline{STBY}$	V_{IH}		V_{CC}-0.5	–	V_{CC}+0.3	V
	EXTAL			V_{CC}x0.7	–	V_{CC}+0.3	
	Other Inputs			2.0	–	V_{CC}+0.3	
Input "Low" Voltage	All Inputs	V_{IL}		-0.3	–	0.8	V
Output "High" Voltage	All Outputs	V_{OH}	I_{OH} = -200μA	2.4	–	–	V
			I_{OH} = -10μA	V_{CC}-0.7	–	–	
Output "Low" Voltage	All Outputs	V_{OL}	I_{OL} = 1.6mA	–	–	0.55	V
Input Leakage Current	TIMER, $\overline{INT}$, $D_1 \sim D_7$, $\overline{STBY}$	$\|I_{IL}\|$	Vin = 0.5 ~ V_{CC}-0.5	–	–	1.0	μA
Three-state Current	$A_0 \sim A_7$, $B_0 \sim B_7$, $C_0 \sim C_7$, $ADR_0 \sim ADR_{13}$*, $DATA_0 \sim DATA_7$, E*, R/$\overline{W}$*	$\|I_{TSI}\|$		–	–	1.0	μA
Current Dissipation**	Operating	I_{CC}	f = 1MHz***	–	5	10	mA
	Wait			–	2	5	mA
	Stop			–	2	10	μA
	Standby			–	2	10	μA
Input Capacitance	All Terminals	Cin	f = 1MHz, Vin = 0V	–	–	12	pF

* Only at standby
** V_{IH} min = V_{CC}-1.0V, V_{IL} max = 0.8V
*** The value at f = x MHz is given by using.
I_{CC} (f = x MHz) = I_{CC} (f = 1MHz) x x

● AC CHARACTERISTICS (V_{CC} = 5.0V±10%, V_{SS} = 0V, Ta = 0 ~ +70°C, unless otherwise noted.)

Item	Symbol	Test Condition	HD6305X2			HD63A05X2			HD63B05X2			Unit
			min	typ	max	min	typ	max	min	typ	max	
Cycle Time	t_{cyc}	Fig. 1	1	–	10	0.666	–	10	0.5	–	10	μs
Enable Rise Time	t_{Er}		–	–	20	–	–	20	–	–	20	ns
Enable Fall Time	t_{Ef}		–	–	20	–	–	20	–	–	20	ns
Enable Pulse Width("High" Level)	PW_{EH}		450	–	–	300	–	–	220	–	–	ns
Enable Pulse Width("Low" Level)	PW_{EL}		450	–	–	300	–	–	220	–	–	ns
Address Delay Time	t_{AD}		–	–	250	–	–	190	–	–	180	ns
Address Hold Time	t_{AH}		40	–	–	30	–	–	20	–	–	ns
Data Delay Time	t_{DW}		–	–	200	–	–	160	–	–	120	ns
Data Hold Time (Write)	t_{HW}		40	–	–	30	–	–	20	–	–	ns
Data Set up Time (Read)	t_{DSR}		80	–	–	60	–	–	50	–	–	ns
Data Hold Time (Read)	t_{HR}		0	–	–	0	–	–	0	–	–	ns

● **PORT TIMING** (V_{CC} = 5.0V±10%, V_{SS} = 0V, Ta = 0 ~ +70°C, unless otherwise noted.)

Item	Symbol	Test Condition	HD6305X2			HD63A05X2			HD63B05X2			Unit
			min	typ	max	min	typ	max	min	typ	max	
Port Data Set-up Time (Port A, B, C, D)	t_{PDS}	Fig. 2	200	–	–	200	–	–	200	–	–	ns
Port Data Hold Time (Port A, B, C, D)	t_{PDH}		200	–	–	200	–	–	200	–	–	ns
Port Data Delay Time (Port A, B, C)	t_{PDW}	Fig. 3	–	–	300	–	–	300	–	–	300	ns

● **CONTROL SIGNAL TIMING** (V_{CC} = 5.0V±10%, V_{SS} = 0V, Ta = 0 ~ +70°C, unless otherwise noted.)

Item	Symbol	Test Condition	HD6305X2			HD63A05X2			HD63B05X2			Unit
			min	typ	max	min	typ	max	min	typ	max	
$\overline{INT}$ Pulse Width	t_{IWL}		t_{cyc} +250	–	–	t_{cyc} +200	–	–	t_{cyc} +200	–	–	ns
$\overline{INT_2}$ Pulse Width	t_{IWL2}		t_{cyc} +250	–	–	t_{cyc} +200	–	–	t_{cyc} +200	–	–	ns
$\overline{RES}$ Pulse Width	t_{RWL}		5	–	–	5	–	–	5	–	–	t_{cyc}
Control Set-up Time	t_{CS}	Fig. 5	250	–	–	250	–	–	250	–	–	ns
Timer Pulse Width	t_{TWL}		t_{cyc} +250	–	–	t_{cyc} +200	–	–	t_{cyc} +200	–	–	ns
Oscillation Start Time (Crystal)	t_{OSC}	Fig.5, Fig.20*	–	–	20	–	–	20	–	–	20	ms
Reset Delay Time	t_{RHL}	Fig. 19	80	–	–	80	–	–	80	–	–	ms

* C_L = 22pF ±20%, R_s = 60Ω max.

● **SCI TIMING** (V_{CC} = 5.0V±10%, V_{SS} = 0V, Ta = 0 ~ +70°C, unless otherwise noted.)

Item	Symbol	Test Condition	HD6305X2			HD63A05X2			HD63B05X2			Unit
			min	typ	max	min	typ	max	min	typ	max	
Clock Cycle	t_{Scyc}	Fig. 6, Fig. 7	1	–	32768	0.67	–	21845	0.5	–	16384	μs
Data Output Delay Time	t_{TXD}		–	–	250	–	–	250	–	–	250	ns
Data Set-up Time	t_{SRX}		200	–	–	200	–	–	200	–	–	ns
Data Hold Time	t_{HRX}		100	–	–	100	–	–	100	–	–	ns

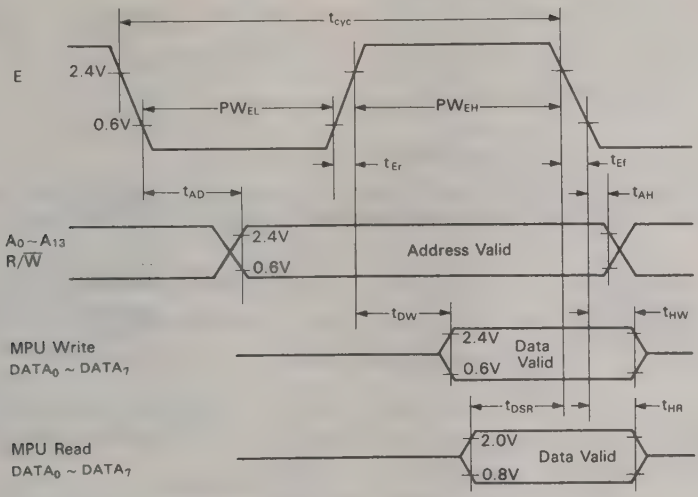


Figure 1 Bus Timing

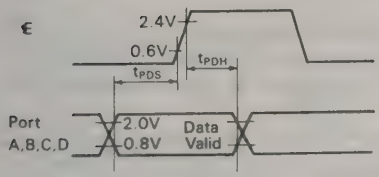


Figure 2 Port Data Set-up and Hold Times (MPU Read)

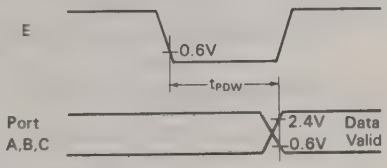


Figure 3 Port Data Delay Time (MPU Write)

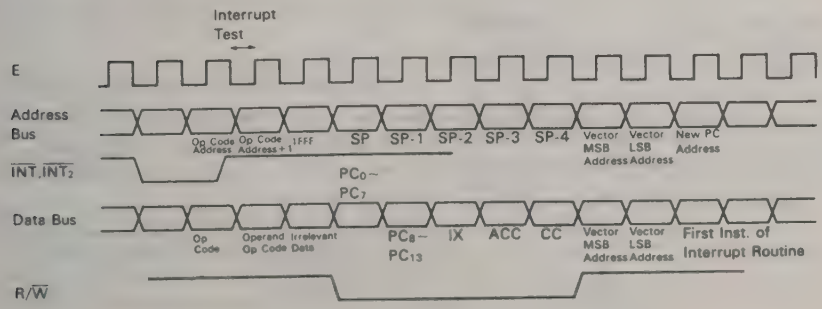


Figure 4 Interrupt Sequence

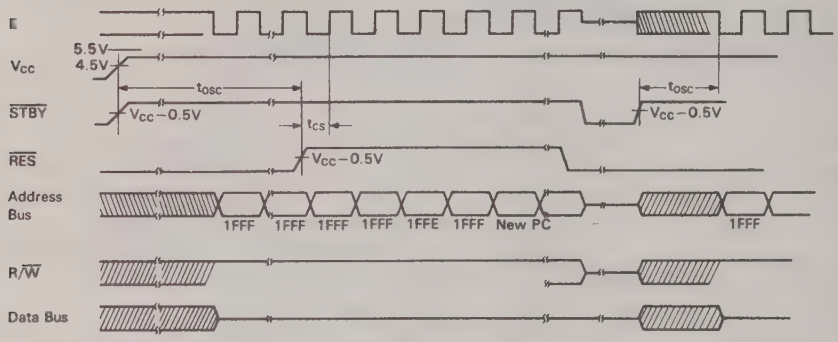


Figure5 Reset Timing

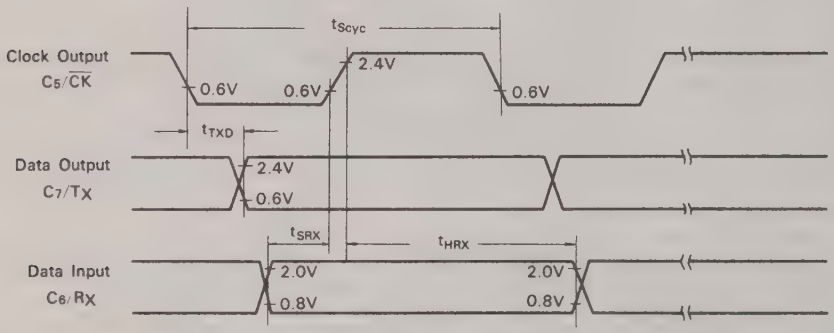


Figure6 SCI Timing (Internal Clock)

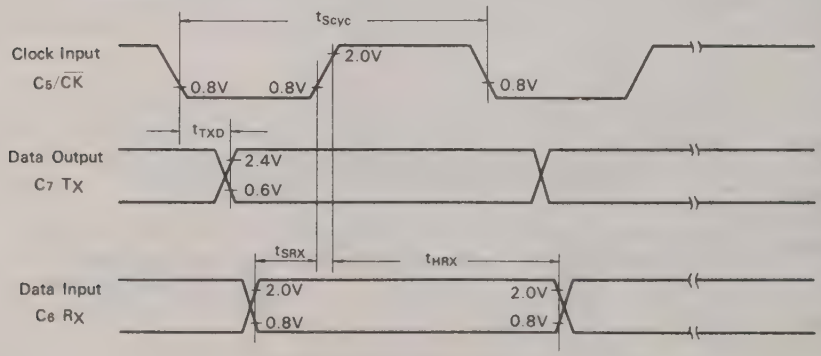
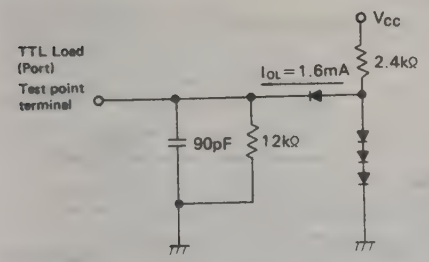


Figure7 SCI Timing(External Clock)



[NOTES] 1. The load capacitance includes stary capacitance caused by the probe, etc.
2. All diodes are 1S2074 Ⓗ.

Figure 8 Test Load

■ DESCRIPTION OF TERMINAL FUNCTIONS

The input and output signals of the MPU are described here.

• V_{CC}, V_{SS}

Voltage is applied to the MPU through these two terminals. V_{CC} is 5.0V ± 10%, while V_{SS} is grounded.

• $\overline{INT}$, $\overline{INT_2}$

External interrupt request inputs to the MPU. For details, refer to "INTERRUPT". The $\overline{INT_2}$ terminal is also used as the port D_6 terminal.

• XTAL, EXTAL

These terminals provide input to the on-chip clock circuit. A crystal oscillator (AT cut, 2.0 to 8.0 MHz) or ceramic filter is connected to the terminal. Refer to "INTERNAL OSCILLATOR" for using these input terminals.

• TIMER

This is an input terminal for event counter. Refer to "TIMER" for details.

• $\overline{RES}$

Used to reset the MPU. Refer to "RESET" for details.

• NUM

This terminal is not for user application. This terminal should be connected to V_{SS}.

• Enable (E)

This output terminal supplies E clock. Output is a single-phase, TTL compatible and 1/4 crystal oscillation frequency or 1/4 external clock frequency. It can drive one TTL load and a 90 pF condenser.

• Read/Write (R/$\overline{W}$)

This TTL compatible output signal indicates to peripheral and memory devices whether MPU is in Read ("High"), or in Write ("Low"). The normal standby state is Read ("High"). Its output can drive one TTL load and a 90pF condenser.

• Data Bus ($DATA_0$ ~ $DATA_7$)

This TTL compatible three-state buffer can drive one TTL load and 90pF.

• Address Bus (ADR_0 ~ ADR_{13})

Each terminal is TTL compatible and can drive one TTL load and 90pF.

• Input/Output Terminals (A_0 ~ A_7, B_0 ~ B_7, C_0 ~ C_7)

These 24 terminals consist of three 8-bit I/O ports (A, B, C). Each of them can be used as an input or output terminal on a bit through program control of the data direction register. For details, refer to "I/O PORTS."

• Input Terminals (D_1 ~ D_7)

These seven input-only terminals are TTL or CMOS compatible. Of the port D's, D_6 is also used as $\overline{INT_2}$. If D_6 is used as a port, the $\overline{INT_2}$ interrupt mask bit of the miscellaneous register must be set to "1" to prevent an $\overline{INT_2}$ interrupt from being accidentally accepted.

• $\overline{STBY}$

This terminal is used to place the MPU into the standby mode. With $\overline{STBY}$ at "Low" level, the oscillation stops and the internal condition is reset. For details, refer to "Stand-by Mode."

The terminals described in the following are I/O pins for serial communication interface (SCI). They are also used as ports C_5, C_6 and C_7. For details, refer to "SERIAL COMMUNICATION INTERFACE."

• $\overline{CK}$ (C_5)

Used to input or output clocks for serial operation.

• Rx (C_6)

Used to receive serial data.

• Tx (C_7)

Used to transmit serial data.

■ MEMORY MAP

The memory map of the MPU is shown in Fig. 9. During interrupt processing, the contents of the CPU registers are saved into the stack in the sequence shown in Fig. 10. This saving begins with the lower byte (PCL) of the program counter. Then the value of the stack pointer is decremented and the higher byte (PCH) of the program counter, index register (X), accumulator (A) and condition code register (CC) are stacked in that order. In a subroutine call, only the contents of the program counter (PCH and PCL) are stacked.

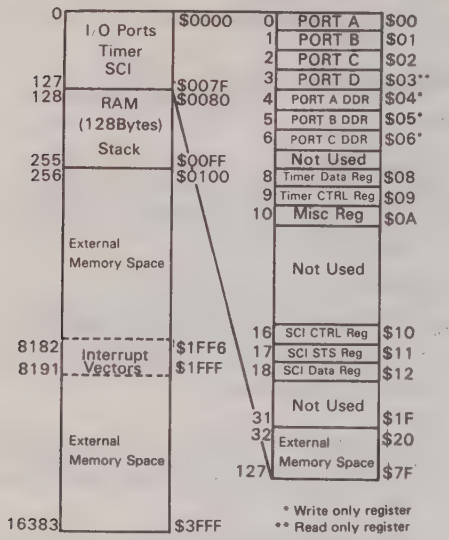


Figure 9 Memory Map of MPU

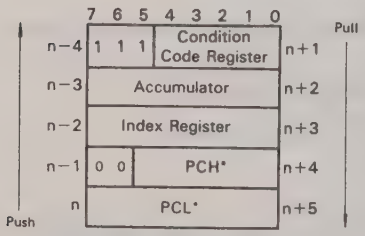

7 6 5 4 3 2 1 0
Pull
n−4 1 1 1 Condition Code Register n+1
n−3 Accumulator n+2
n−2 Index Register n+3
n−1 0 0 PCH* n+4
n PCL* n+5
Push
* In a subroutine call, only PCL and PCH are stacked.

Figure 10 Sequence of Interrupt Stacking

■REGISTERS

There are five registers which the programmer can operate.

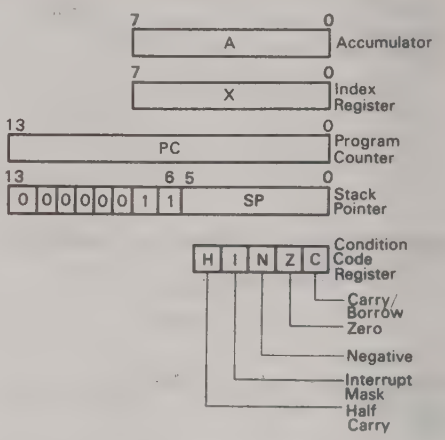


Figure 11 Programming Model

• Accumulator (A)

This accumulator is an ordinary 8-bit register which holds operands or the result of arithmetic operation or data processing.

• Index Register (X)

The index register is an 8-bit register, and is used for index addressing mode. Each of the addresses contained in the register consists of 8 bits which, combined with an offset value, provides an effective address.

In the case of a read/modify/write instruction, the index register can be used like an accumulator to hold operation data or the result of operation.

If not used in the index addressing mode, the register can be used to store data temporarily.

• Program Counter (PC)

The program counter is a 14-bit register that contains the address of the next instruction to be executed.

• Stack Pointer (SP)

The stack pointer is a 14-bit register that indicates the address of the next stacking space. Just after reset, the stack pointer is set at address $00FF. It is decremented when data is pushed, and incremented when pulled. The upper 8 bits of the stack pointer are fixed to 00000011. During the MPU being reset or during a reset stack pointer (RSP) instruction, the pointer is set to address $00FF. Since a subroutine or interrupt can use space up to address $00C1 for stacking, the subroutine can be used up to 31 levels and the interrupt up to 12 levels.

• Condition Code Register (CC)

The condition code register is a 5-bit register, each bit indicating the result of the instruction just executed. The bits can be individually tested by conditional branch instruc-

tions. The CC bits are as follows:

Half Carry (H): Used to indicate that a carry occurred between bits 3 and 4 during an arithmetic operation (ADD, ADC).

Interrupt (I): Setting this bit causes all interrupts, except a software interrupt, to be masked. If an interrupt occurs with the bit I set, it is latched. It will be processed the instant the interrupt mask bit is reset. (More specifically, it will enter the interrupt processing routine after the instruction following the CLI has been executed.)

Negative (N): Used to indicate that the result of the most recent arithmetic operation, logical operation or data processing is negative (bit 7 is logic "1").

Zero (Z): Used to indicate that the result of the most recent arithmetic operation, logical operation or data processing is zero.

Carry/Borrow (C): Represents a carry or borrow that occurred in the most recent arithmetic operation. This bit is also affected by the Bit Test and Branch instruction and a Rotate instruction.

■ INTERRUPT

There are six different types of interrupt: external interrupts ($\overline{INT}$, $\overline{INT_2}$), internal timer interrupts (TIMER, $TIMER_2$), serial interrupt (SCI) and interrupt by an instruction (SWI).

Of these six interrupts, the $\overline{INT_2}$ and TIMER or the SCI and $TIMER_2$ generate the same vector address, respectively.

When an interrupt occurs, the program in progress stops and the then CPU status is saved onto the stack. And then, the interrupt mask bit (I) of the condition code register is set and the start address of the interrupt processing routine is obtained from a particular interrupt vector address. Then the interrupt routine starts from the start address. System can exit from the interrupt routine by an RTI instruction. When this instruction is executed, the CPU status before the interrupt (saved onto the stack) is pulled and the CPU restarts the sequence with the instruction next to the one at which the interrupt occurred. Table 1 lists the priority of interrupts and their vector addresses.

Table 1 Priority of Interrupts

Interrupt	Priority	Vector Address
$\overline{RES}$	1	$1FFE, $1FFF
SWI	2	$1FFC, $1FFD
$\overline{INT}$	3	$1FFA, $1FFB
TIMER/$\overline{INT_2}$	4	$1FF8, $1FF9
SCI/$TIMER_2$	5	$1FF6, $1FF7

A flowchart of the interrupt sequence is shown in Fig. 12. A block diagram of the interrupt request source is shown in Fig. 13.

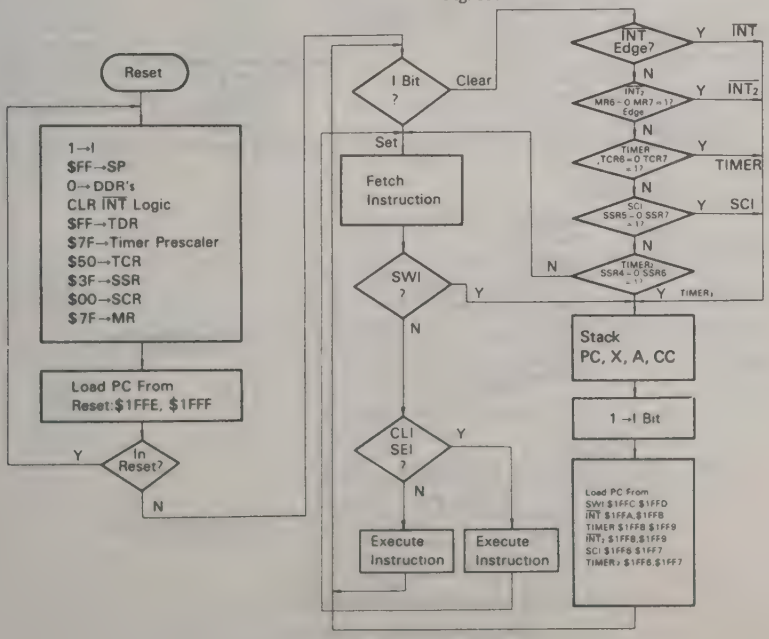


Figure 12 Interrupt Flow Chart

In the block diagram, both the external interrupts $\overline{INT}$ and $\overline{INT_2}$ are edge trigger inputs. At the falling edge of each input, an interrupt request is generated and latched. The $\overline{INT}$ interrupt request is automatically cleared if jumping is made to the $\overline{INT}$ processing routine. Meanwhile, the $\overline{INT_2}$ request is cleared if "0" is written in bit 7 of the miscellaneous register.

For the external interrupts ($\overline{INT}$, $\overline{INT_2}$), internal timer interrupts (TIMER, $TIMER_2$) and serial interrupt (SCI), each interrupt request is held, but not processed, if the I bit of the condition code register is set. Immediately after the I bit is cleared, the corresponding interrupt processing starts according to the priority.

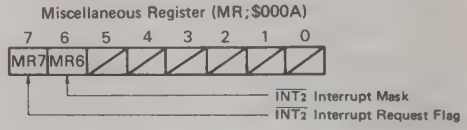
The $\overline{INT_2}$ interrupt can be masked by setting bit 6 of the miscellaneous register; the TIMER interrupt by setting bit 6 of the timer control register; the SCI interrupt by setting bit 5 of the serial status register; and the $TIMER_2$ interrupt by setting bit 4 of the serial status register.

The status of the $\overline{INT}$ terminal can be tested by a BIL or BIH instruction. The $\overline{INT}$ falling edge detector circuit and its latching circuit are independent of testing by these instructions. This is also true with the status of the $\overline{INT_2}$ terminal.

• Miscellaneous Register (MR; $000A)

The interrupt vector address for the external interrupt $\overline{INT_2}$ is the same as that for the TIMER interrupt, as shown in Table 1. For this reason, a special register called the miscellaneous register (MR; $000A) is available to control the $\overline{INT_2}$ interrupts.

Bit 7 of this register is the $\overline{INT_2}$ interrupt request flag. When the falling edge is detected at the $\overline{INT_2}$ terminal, "1" is set in bit 7. Then the software in the interrupt routine (vector addresses: $1FF8, $1FF9) checks bit 7 to see if it is $\overline{INT_2}$ interrupt. Bit 7 can be reset by software.



Miscellaneous Register (MR; $000A)

Bit 6 is the $\overline{INT_2}$ interrupt mask bit. If this bit is set to "1", then the $\overline{INT_2}$ interrupt is disabled. Both read and write are possible with bit 7 but "1" cannot be written in this bit by software. This means that an interrupt request by software is impossible.

When reset, bit 7 is cleared to "0" and bit 6 is set to "1".

■ TIMER

Figure 14 shows a MPU timer block diagram. The timer data register is loaded by software and, upon receipt of a clock input, begins to count down. When the timer data

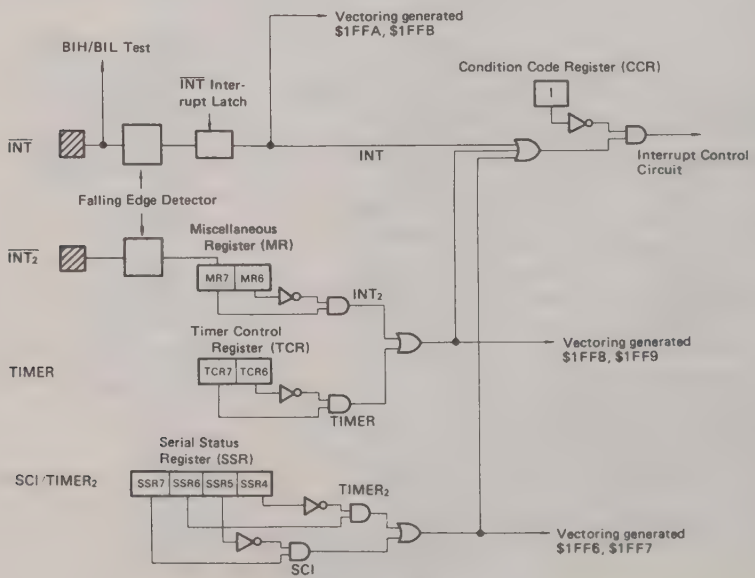


Figure 13 Interrupt Request Generation Circuitry

register (TDR) becomes "0", the timer interrupt request bit (bit 7) in the timer control register is set. In response to the interrupt request, the CPU saves its status into the stack and fetches timer interrupt routine address from addresses $1FF8 and $1FF9 and execute the interrupt routine. The timer interrupt can be masked by setting the timer interrupt mask bit (bit 6) in the timer control register. The mask bit (I) in the condition code register can also mask the timer interrupt.

The source clock to the timer can be either an external signal from the timer input terminal or the internal E signal (the oscillator clock divided by 4). If the E signal is used as the source, the clock input can be gated by the input to the timer input terminal.

Once the timer count has reached "0", it starts counting down with "$FF". The count can be monitored whenever desired by reading the timer data register. This permits the program to know the length of time having passed after the occurrence of a timer interrupt, without disturbing the contents of the counter.

When the MPU is reset, both the prescaler and counter are initialized to logic "1". The timer interrupt request bit (bit 7) then is cleared and the timer interrupt mask bit (bit 6) is set.

To clear the timer interrupt request bit (bit 7), it is necessary to write "0" in that bit.

TCR7	Timer interrupt request
0	Absent
1	Present

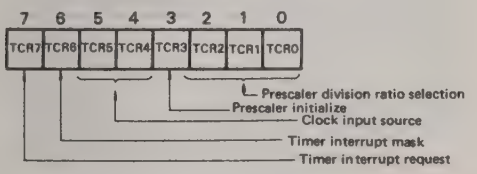
TCR6	Timer interrupt mask
0	Enabled
1	Disabled

• **Timer Control Register (TCR; $0009)**

Selection of a clock source, selection of a prescaler frequency division ratio, and a timer interrupt can be controlled by the timer control register (TCR; $0009).

For the selection of a clock source, any one of the four modes (see Table 2) can be selected by bits 5 and 4 of the timer control register (TCR).

Timer Control Register (TCR; $0009)


7 6 5 4 3 2 1 0
TCR7 TCR6 TCR5 TCR4 TCR3 TCR2 TCR1 TCR0
Prescaler division ratio selection
Prescaler initialize
Clock input source
Timer interrupt mask
Timer interrupt request

After reset, the TCR is initialized to "E under timer terminal control" (bit 5 = 0, bit 4 = 1). If the timer terminal is "1", the counter starts counting down with "$FF" immediately after reset.

When "1" is written in bit 3, the prescaler is initialized. This bit always shows "0" when read.

Table 2 Clock Source Selection

TCR		Clock input source
Bit 5	Bit 4	
0	0	Internal clock E
0	1	E under timer terminal control
1	0	No clock input (counting stopped)
1	1	Event input from timer terminal

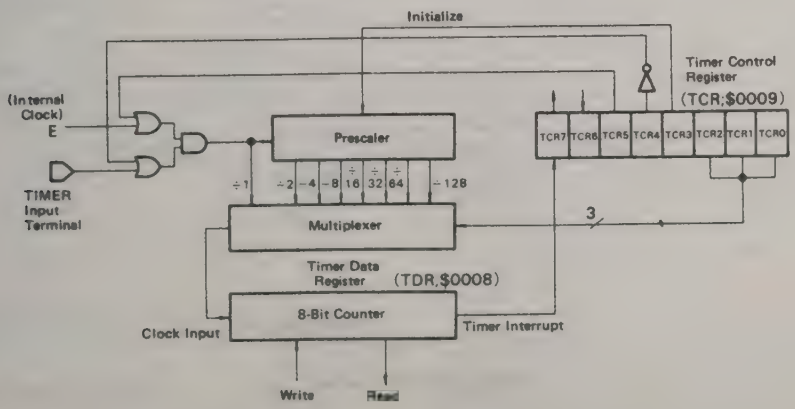


Figure 14 Timer Block Diagram

A prescaler division ratio is selected by the combination of three bits (bits 0, 1 and 2) of the timer control register (see Table 3). There are eight different division ratios: ÷1, ÷2, ÷4, ÷8, ÷16, ÷32, ÷64 and ÷128. After reset, the TCR is set to the ÷1 mode.

Table 3 Prescaler Division Ratio Selection

TCR			Prescaler division ratio
Bit 2	Bit 1	Bit 0	
0	0	0	÷1
0	0	1	÷2
0	1	0	÷4
0	1	1	÷8
1	0	0	÷16
1	0	1	÷32
1	1	0	÷64
1	1	1	÷128

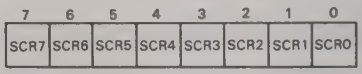
A timer interrupt is enabled when the timer interrupt mask bit is "0", and disabled when the bit is "1". When a timer interrupt occurs, "1" is set in the timer interrupt request bit. This bit can be cleared by writing "0" in that bit.

■SERIAL COMMUNICATION INTERFACE (SCI)

This interface is used for serial transmission or reception of 8-bit data. Sixteen transfer rates are available in the range from 1 μs to approx. 32 ms (for oscillation at 4 MHz).

The SCI consists of three registers, one eighth counter and one prescaler. (See Fig. 15.) SCI communicates with the CPU via the data bus, and with the outside world through bits 5, 6 and 7 of port C. Described below are the operations of each register and data transfer.

●SCI Control Register (SCR; $0010)



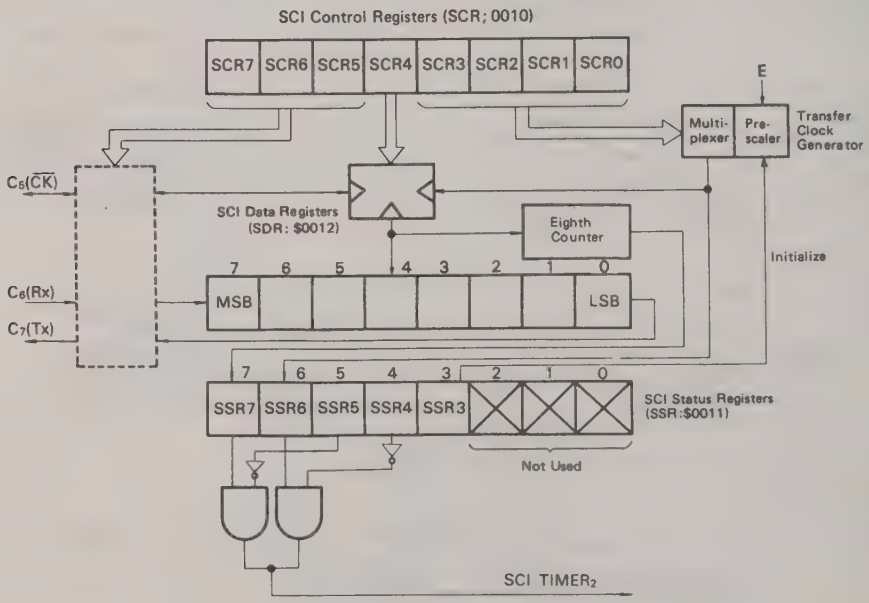


Figure 15 SCI Block Diagram

SCR7	C_7 terminal
0	Used as I/O terminal (by DDR).
1	Serial data output (DDR output)

SCR6	C_6 terminal
0	Used as I/O terminal (by DDR).
1	Serial data input (DDR input)

SCR5	SCR4	Clock source	C_5 terminal
0	0	–	Used as I/O terminal (by DDR).
0	1	–	
1	0	Internal	Clock output (DDR output)
1	1	External	Clock input (DDR input)

Bit 7 (SCR7)

When this bit is set, the DDR corresponding to the C_7 becomes "1" and this terminal serves for output of SCI data. After reset, the bit is cleared to "0".

Bit 6 (SCR6)

When this bit is set, the DDR corresponding to the C_6 becomes "0" and this terminal serves for input of SCI data. After reset, the bit is cleared to "0".

Bits 5 and 4 (SCR5, SCR4)

These bits are used to select a clock source. After reset, the bits are cleared to "0".

Bits 3 ~ 0 (SCR3 ~ SCR0)

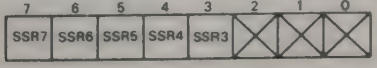
These bits are used to select a transfer clock rate. After reset, the bits are cleared to "0".

SCR3	SCR2	SCR1	SCR0	Transfer clock rate	
				4.00 MHz	4.194 MHz
0	0	0	0	1 μs	0.95 μs
0	0	0	1	2 μs	1.91 μs
0	0	1	0	4 μs	3.82 μs
0	0	1	1	8 μs	7.64 μs
≀	≀	≀	≀	≀	≀
1	1	1	1	32768 μs	1/32 s

•SCI Data Register (SDR; $0012)

A serial-parallel conversion register that is used for transfer of data.

•SCI Status Register (SSR; $0011)



Bit 7 (SSR7)

Bit 7 is the SCI interrupt request bit which is set upon completion of transmitting or receiving 8-bit data. It is cleared when reset or data is written to or read from the SCI data register with the SCR5="1". The bit can also be cleared by writing "0" in it.

Bit 6 (SSR6)

Bit 6 is the $TIMER_2$ interrupt request bit. $TIMER_2$ is used commonly with the serial clock generator, and SSR6 is set each time the internal transfer clock falls. When reset, the bit is cleared. It also be cleared by writing "0" in it. (For details, see $TIMER_2$.)

Bit 5 (SSR5)

Bit 5 is the SCI interrupt mask bit which can be set or cleared by software. When it is "1", the SCI interrupt (SSR7) is masked. When reset, it is set to "1".

Bit 4 (SSR4)

Bit 4 is the $TIMER_2$ interrupt mask bit which can be set or cleared by software. When the bit is "1", the $TIMER_2$ interrupt (SSR6) is masked. When reset, it is set to "1".

Bit 3 (SSR3)

When "1" is written in this bit, the prescaler of the transfer clock generator is initialized. When read, the bit always is "0".

Bits 2 ~ 0

Not used.

SSR7	SCI interrupt request
0	Absent
1	Present

SSR6	$TIMER_2$ interrupt request
0	Absent
1	Present

SSR5	SCI interrupt mask
0	Enabled
1	Disabled

SSR4	$TIMER_2$ interrupt mask
0	Enabled
1	Disabled

• Data Transmission

By writing the desired control bits into the SCI control registers, a transfer rate and a source of transfer clock are determined and bits 7 and 5 of port C are set at the serial data output terminal and the serial clock terminal, respectively. The transmit data should be stored from the accumulator or index register into the SCI data register. The data written in the SCI data register is output from the C_7/Tx terminal, starting with the LSB, synchronously with the falling edge of the serial clock. (See Fig. 16.) When 8 bit of

data have been transmitted, the interrupt request bit is set in bit 7 of the SCI status register with the rising edge of the last serial clock. This request can be masked by setting bit 5 of the SCI status register. Once the data has been sent, the 8th bit data (MSB) stays at the C_7/Tx terminal. If an external clock source has been selected, the transfer rate determined by bits 0 ~ 3 of the SCI control register is ignored, and the $C_5/\overline{CK}$ terminal is set as input. If the internal clock has been selected, the $C_5/\overline{CK}$ terminal is set as output and clocks are output at the transfer rate selected by bits 0 ~ 3 of the SCI control register.

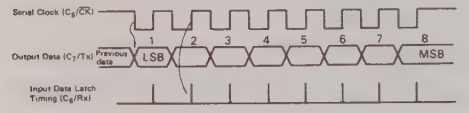


Figure 16 SCI Timing Chart

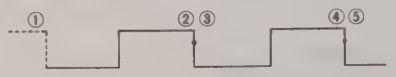
• Data Reception

By writing the desired control bits into the SCI control register, a transfer rate and a source of transfer clock are determined and bit 6 and 5 of port C are set at the serial data input terminal and the serial clock terminal, respectively. Then dummy-writing or -reading the SCI data register, the system is ready for receiving data. (This procedure is not needed after reading the subsequent received data. It must be taken after reset and after not reading the subsequent received data.)

The data from the C_6/Rx terminal is input to the SCI data register synchronously with the rising edge of the serial clock (see Fig. 16). When 8 bits of data have been received, the interrupt request bit is set in bit 7 of the SCI status register. This request can be masked by setting bit 5 of the SCI status register. If an external clock source have been selected, the transfer rate determined by bits 0 ~ 3 of the SCI control register is ignored and the data is received synchronously with the clock from the $C_5/\overline{CK}$ terminal. If the internal clock has been selected, the $C_5/\overline{CK}$ terminal is set as output and clocks are output at the transfer rate selected by bits 0 ~ 3 of the SCI control register.

• TIMER$_2$

The SCI transfer clock generator can be used as a timer. The clock selected by bits 3 ~ 0 of the SCI control register (4μs ~ approx. 32 ms (for oscillation at 4 MHz)) is input to bit 6 of the SCI status register and the TIMER$_2$ interrupt request bit is set at each falling edge of the clock. Since interrupt requests occur periodically, TIMER$_2$ can be used as a reload counter or clock.



① : Transfer clock generator is reset and mask bit (bit 4 of SCI status register) is cleared.
②, ④ : TIMER$_2$ interrupt request
③, ⑤ : TIMER$_2$ interrupt request bit cleared

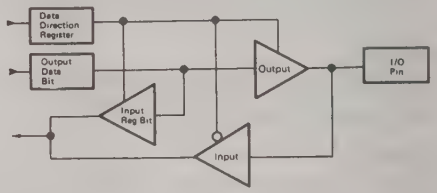
TIMER$_2$ is commonly used with the SCI transfer clock generator. If wanting to use TIMER$_2$ independently of the SCI, specify "External" (SCR5 = 1, SCR4 = 1) as the SCI clock source.

If "Internal" is selected as the clock source, reading or writing the SDR causes the prescaler of the transfer clock generator to be initialized.

■ I/O PORTS

There are 24 input/output terminals (ports A, B, C). Each I/O terminal can be selected for either input or output by the data direction register. More specifically, an I/O port will be input if "0" is written in the data direction register, and output if "1" is written in the data direction register. Port A, B or C reads latched data if it has been programmed as output, even with the output level being fluctuated by the output load. (See Fig. 17.)

When reset, the data direction register and data register go to "0" and all the input/output terminals are used as input.



Bit of data direction register	Bit of output data	Status of output	Input to CPU
1	0	0	0
1	1	1	1
0	X	3-state	Pin

Figure 17 Input/Output Port Diagram

Seven input-only terminals are available (port D). Writing to an input terminal is invalid.

All input/output terminals and input terminals are TTL compatible and CMOS compatible in respect of both input and output.

If I/O ports or input ports are not used, they should be connected to V_{SS} via resistors. With none connected to these terminals, there is the possibility of power being consumed despite that they are not used.

■ RESET

The MPU can be reset either by external reset input ($\overline{RES}$) or power-on reset. (See Fig. 18.) On power up, the reset input must be held "Low" for at least t_{OSC} to assure that the internal oscillator is stabilized. A sufficient time of delay can be obtained by connecting a capacitance to the $\overline{RES}$ input as shown in Fig. 19.

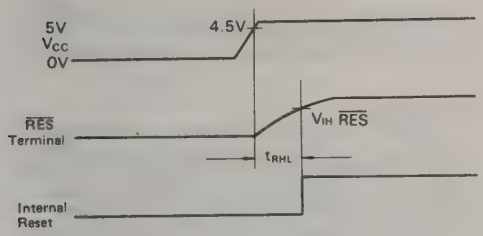


Figure 18 Power On and Reset Timing

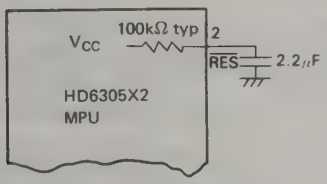


Figure 19 Input Reset Delay Circuit

■INTERNAL OSCILLATOR

The internal oscillator circuit is designed to meet the requirement for minimum external configurations. It can be driven by connecting a crystal (AT cut 2.0 ~ 8.0MHz) or ceramic oscillator between pins 5 and 6 depending on the required oscillation frequency stability.

Three different terminal connections are shown in Fig. 20. Figs. 21 and 22 illustrate the specifications and typical arrangement of the crystal, respectively.

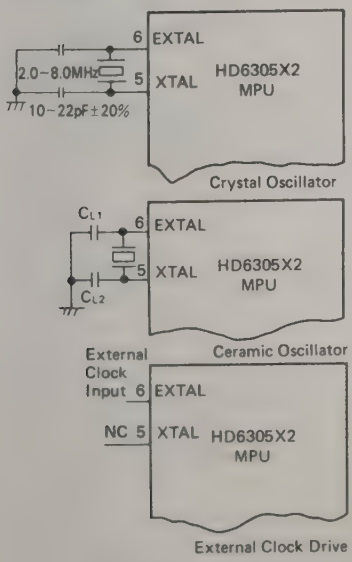


Figure 20 Internal Oscillator Circuit

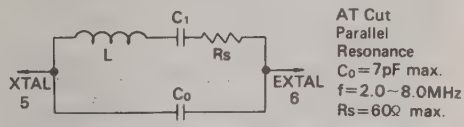
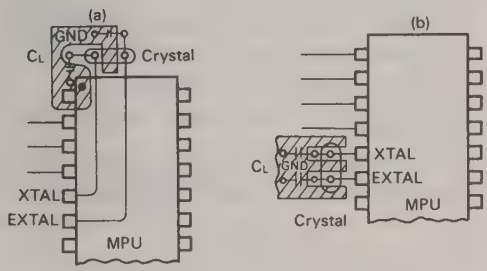


Figure 21 Parameters of Crystal



[NOTE] Use as short wirings as possible for connection of the crystal with the EXTAL and XTAL terminals. Do not allow these wirings to cross others.

Figure 22 Typical Crystal Arrangement

■LOW POWER DISSIPATION MODE

The HD6305X2 has three low power dissipation modes: wait, stop and standby.

• Wait Mode

When WAIT instruction being executed, the MPU enters into the wait mode. In this mode, the oscillator stays active but the internal clock stops. The CPU stops but the peripheral functions – the timer and the serial communication interface – stay active. (NOTE: Once the system has entered the wait mode, the serial communication interface can no longer be retriggered.) In the wait mode, the registers, RAM and I/O terminals hold their condition just before entering into the wait mode.

The escape from this mode can be done by interrupt ($\overline{INT}$, TIMER/$\overline{INT_2}$ or SCI/TIMER$_2$), $\overline{RES}$ or $\overline{STBY}$. The $\overline{RES}$ resets the MPU and the $\overline{STBY}$ brings it into the standby mode. (This will be mentioned later.)

When interrupt is requested to the CPU and accepted, the wait mode escapes, then the CPU is brought to the operation mode and vectors to the interrupt routine. If the interrupt is masked by the I bit of the condition code register, after releasing from the wait mode the MPU executes the instruction next to the WAIT. If an interrupt other than the $\overline{INT}$ (i.e., TIMER/$\overline{INT_2}$ or SCI/TIMER$_2$) is masked by the timer control

register, miscellaneous register or serial status register, there is no interrupt request to the CPU, so the wait mode cannot be released.

Fig. 23 shows a flowchart for the wait function.

• Stop Mode

When STOP instruction being executed, MPU enters into the stop mode. In this mode, the oscillator stops and the CPU and peripheral functions become inactive but the RAM, registers and I/O terminals hold their condition just before entering into the stop mode.

The escape from this mode can be done by an external interrupt ($\overline{INT}$ or $\overline{INT_2}$), $\overline{RES}$ or $\overline{STBY}$. The $\overline{RES}$ resets the MPU and the $\overline{STBY}$ brings into the standby mode.

When interrupt is requested to the CPU and accepted, the stop mode escapes, then the CPU is brought to the operation mode and vectors to the interrupt routine. If the interrupt is masked by the I bit of the condition code register, after releasing from the stop mode, the MPU executes the instruction next to the STOP. If the $\overline{INT_2}$ interrupt is masked by the miscellaneous register, there is no interrupt request to the MPU, so the stop mode cannot be released.

Fig. 24 shows a flowchart for the stop function. Fig. 25 shows a timing chart of return to the operation mode from the stop mode.

For releasing from the stop mode by an interrupt, oscillation starts upon input of the interrupt and, after the internal delay time for stabilized oscillation, the CPU becomes active.

For restarting by $\overline{RES}$, oscillation starts when the $\overline{RES}$ goes "0" and the CPU restarts when the $\overline{RES}$ goes "1". The duration of $\overline{RES}$="0" must exceed t_{osc} to assure stabilized oscillation.

• Standby Mode

The MPU enters into the standby mode when the $\overline{STBY}$ terminal goes "Low". In this mode, all operations stop and the internal condition is reset but the contents of the RAM are hold. The I/O terminals turn to high-impedance state. The standby mode should escape by bringing $\overline{STBY}$ "High". The CPU must be restarted by reset. The timing of input signals at the $\overline{RES}$ and $\overline{STBY}$ terminals is shown in Fig. 26.

Table 4 lists the status of each parts of the MPU in each low power dissipation modes. Transitions between each mode are shown in Fig. 27.

(Note) When I bit of condition code register is "1" and interrupt ($\overline{INT}$, TIMER/$\overline{INT_2}$, SCI/$TIMER_2$) is held, MPU does not enter WAIT mode by the execution of WAIT instruction.
In that case, after the 4 dummy cycles MCU executes the next instruction.
In the same way, when external interrupts ($\overline{INT}$, $\overline{INT_2}$) are held at the bit I set, MPU does not enter STOP mode by the execution of STOP instruction. In that case, also, MPU executes the next instruction after the 4 dummy cycles.

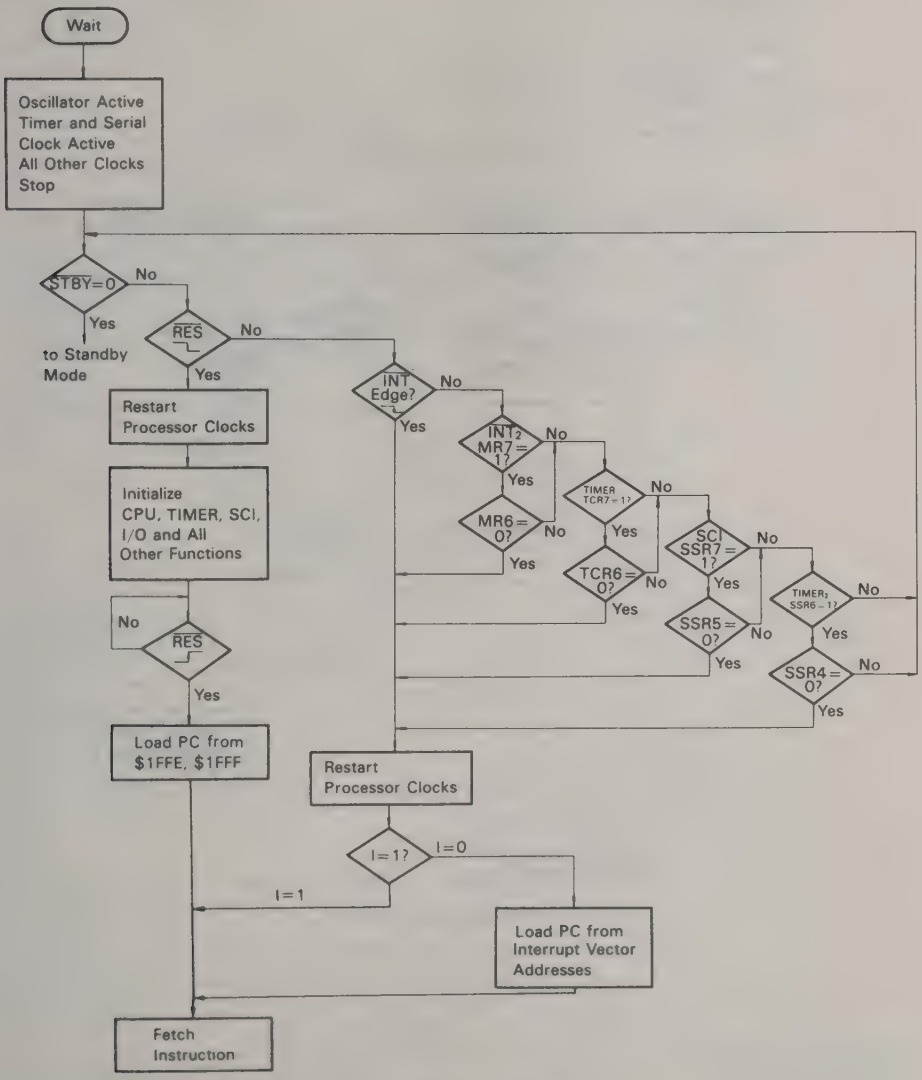


Figure 23 Wait Mode Flow Chart

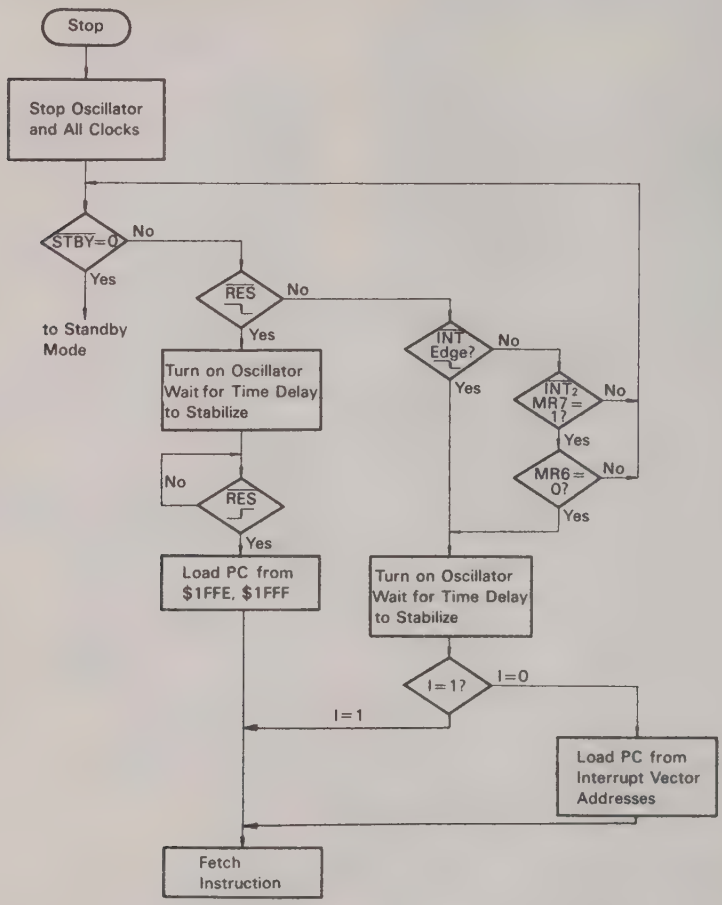
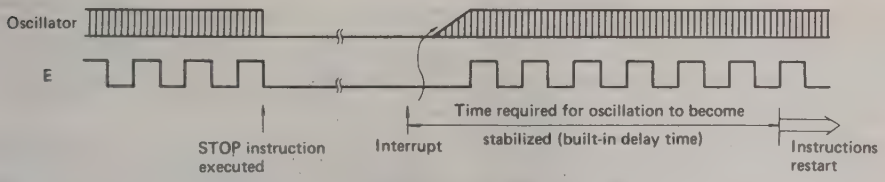
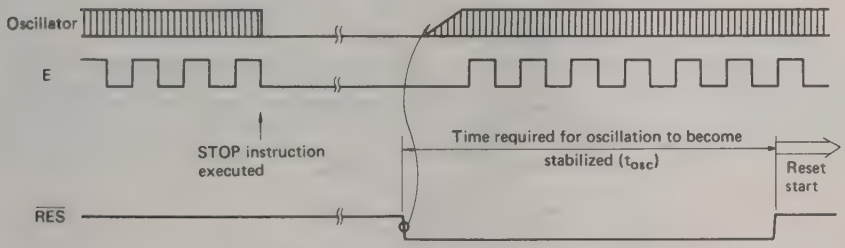


Figure 24 Stop Mode Flow Chart



(a) Restart by Interrupt



(b) Restart by Reset

Figure 25 Timing Chart of Releasing from Stop Mode

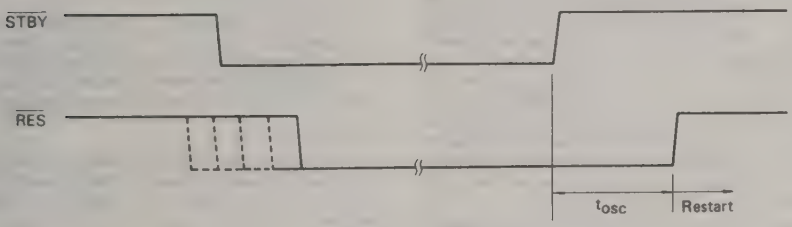


Figure 26 Timing Chart of Releasing from Standby Mode

Table 4 Status of Each Part of MPU in Low Power Dissipation Modes

Mode	Start		Condition						Escape
			Oscillator	CPU	Timer, Serial	Register	RAM	I/O terminal	
WAIT	Software	WAIT instruction	Active	Stop	Active	Keep	Keep	Keep	$\overline{STBY}$, $\overline{RES}$, $\overline{INT}$, $\overline{INT_2}$, each interrupt request of TIMER, $TIMER_2$, SCI
STOP		STOP instruction	Stop	Stop	Stop	Keep	Keep	Keep	$\overline{STBY}$, $\overline{RES}$, $\overline{INT}$, $\overline{INT_2}$
Standby	Hardware	$\overline{STBY}$="Low"	Stop	Stop	Stop	Reset	Keep	High impedance	$\overline{STBY}$="High"

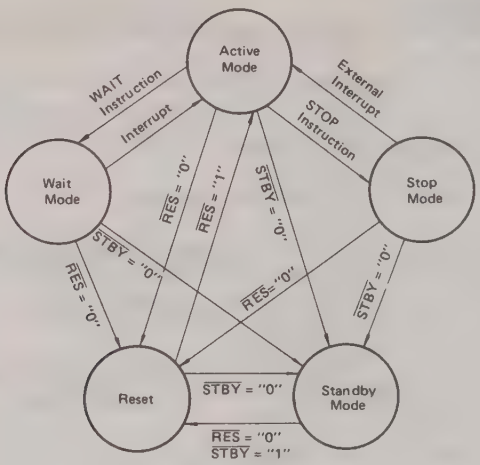


Figure 27 Transitions among Active Mode, Wait Mode, Stop Mode, Standby Mode and Reset

■BIT MANIPULATION

The MPU can use a single instruction (BSET or BCLR) to set or clear one bit of the RAM or an I/O port (except the write-only registers such as the data direction register). Every bit of memory or I/O within page 0 ($00 ~ $FF) can be tested by the BRSET or BRCLR instruction; depending on the result of the test, the program can branch to required destinations. Since bits in the RAM, or I/O can be manipulated, the user may use a bit within the RAM as a flag or handle a single I/O bit as an independent I/O terminal. Fig. 28 shows an example of bit manipulation and the validity of test instructions. In the example, the program is configured assuming that bit 0 of port A is connected to a zero cross detector circuit and bit 1 of the same port to the trigger of a triac.

The program shown can activate the triac within a time of 10μs from zero-crossing through the use of only 7 bytes on the memory. The on-chip timer provides a required time of delay and pulse width modulation of power is also possible.

```
        :
SELF 1. BRCLR 0, PORT A, SELF 1
        BSET 1, PORT A
        BCLR 1, PORT A
        :
```

Figure 28 Example of Bit Manipulation

■ADDRESSING MODES

Ten different addressing modes are available to the MPU.

• Immediate

See Fig. 29. The immediate addressing mode provides access to a constant which does not vary during execution of the program.

This access requires an instruction length of 2 bytes. The effective address (EA) is PC and the operand is fetched from the byte that follows the operation code.

• Direct

See Fig. 30. In the direct addressing mode, the address of the operand is contained in the 2nd byte of the instruction. The user can gain direct access to memory up to the lower 255th address. All RAM and I/O registers are on page 0 of address space so that the direct addressing mode may be utilized.

• Extended

See Fig. 31. The extended addressing is used for referencing to all addresses of memory. The EA is the contents of the 2 bytes that follow the operation code. An extended addressing instruction requires a length of 3 bytes.

• Relative

See Fig. 32. The relative addressing mode is used with branch instructions only. When a branch occurs, the program counter is loaded with the contents of the byte following the operation code. EA = (PC) + 2 + Rel., where Rel. indicates a signed 8-bit data following the operation code. If no branch occurs, Rel. = 0. When a branch occurs, the program jumps to any byte in the range +129 to −127. A branch instruction requires a length of 2 bytes.

• Indexed (No Offset)

See Fig. 33. The indexed addressing mode allows access up to the lower 255th address of memory. In this mode, an instruction requires a length of one byte. The EA is the contents of the index register.

• Indexed (8-bit Offset)

See Fig. 34. The EA is the contents of the byte following the operation code, plus the contents of the index register. This mode allows access up to the lower 511th address of memory. Each instruction when used in the index addressing mode (8-bit offset) requires a length of 2 bytes.

• Indexed (16-bit Offset)

See Fig. 35. The contents of the 2 bytes following the operation code are added to content of the index register to compute the value of EA. In this mode, the complete memory can be accessed. When used in the indexed addressing mode (16-bit offset), an instruction must be 3 bytes long.

• Bit Set/Clear

See Fig. 36. This addressing mode is applied to the BSET and BCLR instructions that can set or clear any bit on page 0. The lower 3 bits of the operation code specify the bit to be set or cleared. The byte that follows the operation code indicates an address within page 0.

• Bit Test and Branch

See Fig. 37. This addressing mode is applied to the BRSET and BRCLR instructions that can test any bit within page 0 and can be branched in the relative addressing mode. The byte to be tested is addressed depending on the contents of the byte following the operation code. Individual bits within the byte to be tested are specified by the lower 3 bits of the operation code. The 3rd byte represents a relative value which will be added to the program counter when a branch condition is established. Each of these instructions should be 3 bytes long. The value of the test bit is written in the carry bit of the condition code register.

• Implied

See Fig. 38. This mode involves no EA. All information needed for execution of an instruction is contained in the operation code. Direct manipulation on the accumulator and index register is included in the implied addressing mode. Other instructions such as SWI and RTI are also used in this mode. All instructions used in the implied addressing mode should have a length of one byte.

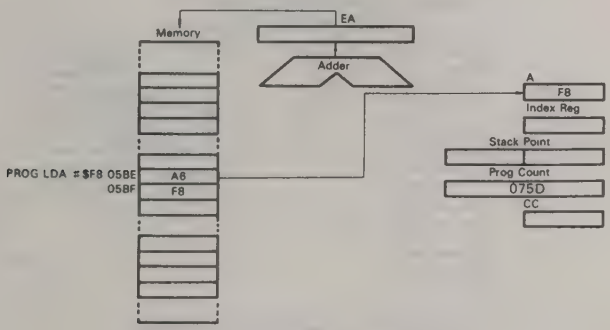


Figure 29 Example of Immediate Addressing

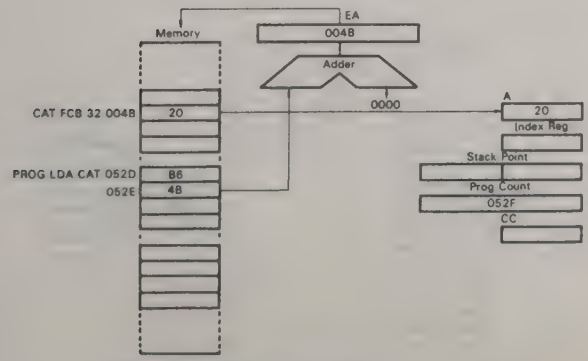


Figure 30 Example of Direct Addressing

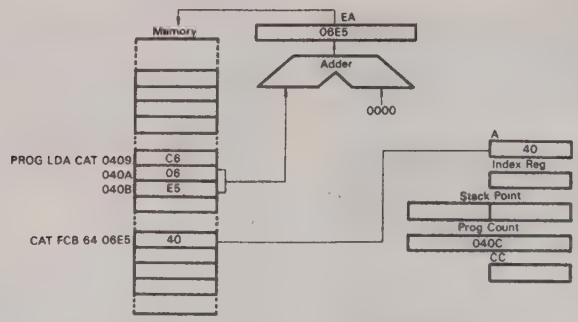


Figure 31 Example of Extended Addressing

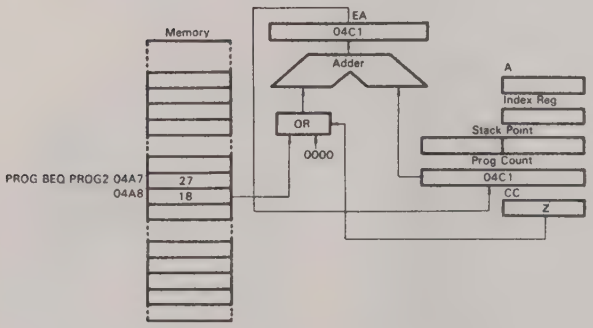


Figure 32 Example of Relative Addressing

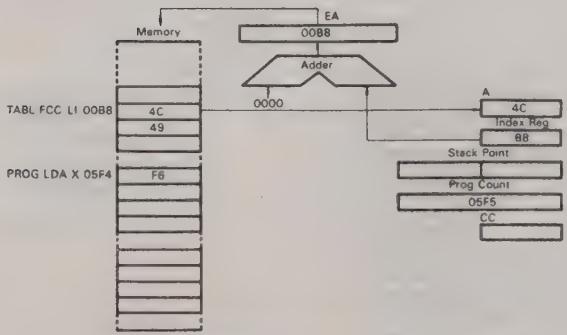


Figure 33 Example of Indexed (No Offset) Addressing

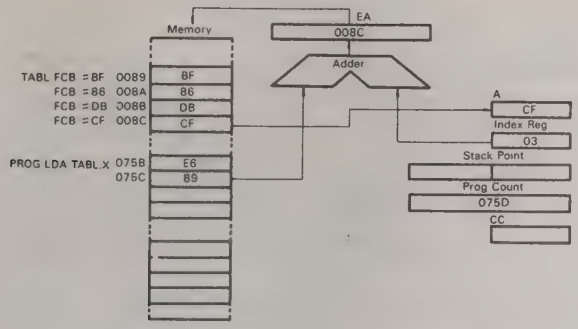


Figure 34 Example of Index (8-bit Offset) Addressing

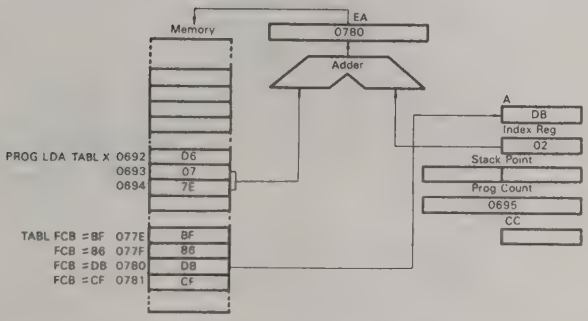


Figure 35 Example of Index (16-bit Offset) Addressing

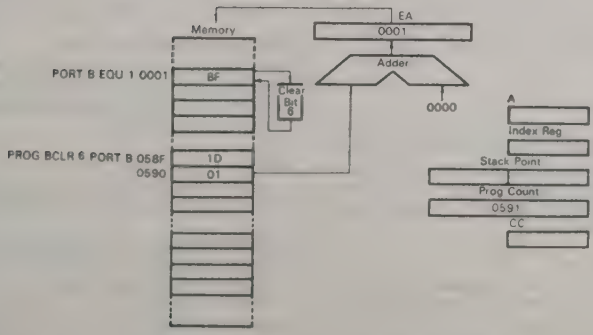


Figure 36 Example of Bit Set/Clear Addressing

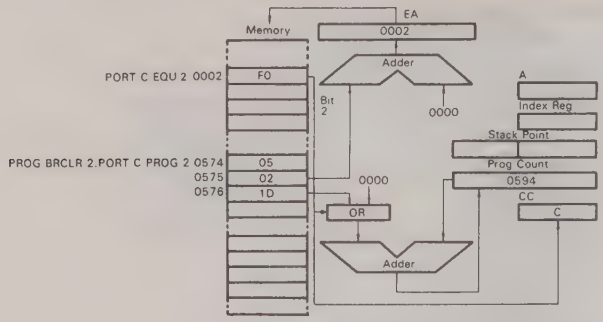


Figure 37 Example of Bit Test and Branch Addressing

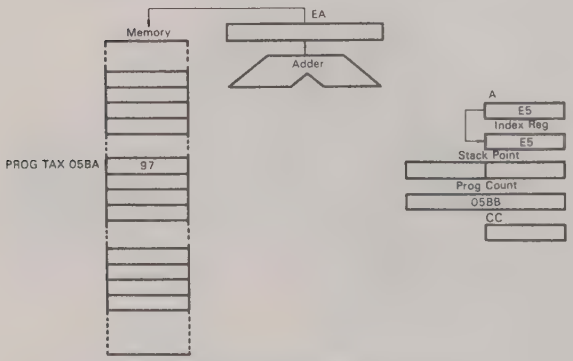


Figure 38 Example of Implied Addressing

■INSTRUCTION SET

There are 62 basic instructions available to the HD6305X2 MPU. They can be classified into five categories: register/memory, read/modify/write, branch, bit manipulation, and control. The details of each instruction are described in Tables 5 through 11.

• Register/Memory Instructions

Most of these instructions use two operands. One operand is either an accumulator or index register. The other is derived from memory using one of the addressing modes used on the HD6305X2 MPU. There is no register operand in the unconditional jump instruction (JMP) and the subroutine jump instruction (JSR). See Table 5.

• Read/Modify/Write Instructions

These instructions read a memory or register, then modify or test its contents, and write the modified value into the memory or register. Zero test instruction (TST) does not write data, and is handled as an exception in the read/modify/write group. See Table 6.

• Branch Instructions

A branch instruction branches from the program sequence in progress if a particular condition is established. See Table 7.

• Bit Manipulation Instructions

These instructions can be used with any bit located up to the lower 255th address of memory. Two groups are available; one for setting or clearing and the other for bit testing and branching. See Table 8.

• Control Instructions

The control instructions control the operation of the MPU which is executing a program. See Table 9.

• List of Instructions in Alphabetical Order

Table 10 lists all the instructions used on the HD6305X2 MPU in the alphabetical order.

• Operation Code Map

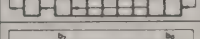
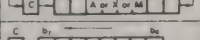
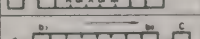
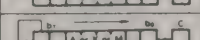
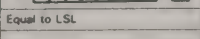
Table 11 shows the operation code map for the instructions used on the MPU.

Table 5 Register/Memory Instructions

Operations	Mnemonic	Addressing Modes: Immediate OP	#	~	Direct OP	#	~	Extended OP	#	~	Indexed (No Offset) OP	#	~	Indexed (8-Bit Offset) OP	#	~	Indexed (16-Bit Offset) OP	#	~	Boolean/ Arithmetic Operation	Condition Code H	I	N	Z	C
Load A from Memory	LDA	A6	2	2	B6	2	3	C6	3	4	F6	1	3	E6	2	4	D6	3	5	M→A	•	•	^	^	•
Load X from Memory	LDX	AE	2	2	BE	2	3	CE	3	4	FE	1	3	EE	2	4	DE	3	5	M→X	•	•	^	^	•
Store A in Memory	STA				B7	2	3	C7	3	4	F7	1	4	E7	2	4	D7	3	5	A→M	•	•	^	^	•
Store X in Memory	STX				BF	2	3	CF	3	4	FF	1	4	EF	2	4	DF	3	5	X→M	•	•	^	^	•
Add Memory to A	ADD	AB	2	2	BB	2	3	CB	3	4	FB	1	3	EB	2	4	DB	3	5	A+M→A	^	•	^	^	^
Add Memory and Carry to A	ADC	A9	2	2	B9	2	3	C9	3	4	F9	1	3	E9	2	4	D9	3	5	A+M+C→A	^	•	^	^	^
Subtract Memory	SUB	A0	2	2	B0	2	3	C0	3	4	F0	1	3	E0	2	4	D0	3	5	A−M →A	•	•	^	^	^
Subtract Memory from A with Borrow	SBC	A2	2	2	B2	2	3	C2	3	4	F2	1	3	E2	2	4	D2	3	5	A−M−C →A	•	•	^	^	^
AND Memory to A	AND	A4	2	2	B4	2	3	C4	3	4	F4	1	3	E4	2	4	D4	3	5	A · M→A	•	•	^	^	•
OR Memory with A	ORA	AA	2	2	BA	2	3	CA	3	4	FA	1	3	EA	2	4	DA	3	5	A+M→A	•	•	^	^	•
Exclusive OR Memory with A	EOR	A8	2	2	B8	2	3	C8	3	4	F8	1	3	E8	2	4	D8	3	5	A⊕M→A	•	•	^	^	•
Arithmetic Compare A with Memory	CMP	A1	2	2	B1	2	3	C1	3	4	F1	1	3	E1	2	4	D1	3	5	A−M	•	•	^	^	^
Arithmetic Compare X with Memory	CPX	A3	2	2	B3	2	3	C3	3	4	F3	1	3	E3	2	4	D3	3	5	X−M	•	•	^	^	^
Bit Test Memory with A (Logical Compare)	BIT	A5	2	2	B5	2	3	C5	3	4	F5	1	3	E5	2	4	D5	3	5	A · M	•	•	^	^	•
Jump Unconditional	JMP	–	–	–	BC	2	2	CC	3	3	FC	1	2	EC	2	3	DC	3	4		•	•	•	•	•
Jump to Subroutine	JSR	–	–	–	BD	2	5	CD	3	6	FD	1	5	ED	2	5	DD	3	6		•	•	•	•	•

Symbols: Op = Operation
= Number of bytes
~ = Number of cycles

Table 6 Read/Modify/Write Instructions

Operations	Mnemonic	Addressing Modes: Implied(A) OP	#	~	Implied(X) OP	#	~	Direct OP	#	~	Indexed (No Offset) OP	#	~	Indexed (8-Bit Offset) OP	#	~	Boolean/Arithmetic Operation	Condition Code H	I	N	Z	C
Increment	INC	4C	1	2	5C	1	2	3C	2	5	7C	1	5	6C	2	6	A+1→A or X+1→X or M+1→M	•	•	^	^	•
Decrement	DEC	4A	1	2	5A	1	2	3A	2	5	7A	1	5	6A	2	6	A−1→A or X−1→X or M−1→M	•	•	^	^	•
Clear	CLR	4F	1	2	5F	1	2	3F	2	5	7F	1	5	6F	2	6	00→A or 00→X or 00→M	•	•	0	1	•
Complement	COM	43	1	2	53	1	2	33	2	5	73	1	5	63	2	6	$\overline{A}$ →A or $\overline{X}$→X or $\overline{M}$ →M	•	•	^	^	1
Negate (2's Complement)	NEG	40	1	2	50	1	2	30	2	5	70	1	5	60	2	6	00−A→A or 00−X→X or 00−M→M	•	•	^	^	^
Rotate Left Thru Carry	ROL	49	1	2	59	1	2	39	2	5	79	1	5	69	2	6		•	•	^	^	^
Rotate Right Thru Carry	ROR	46	1	2	56	1	2	36	2	5	76	1	5	66	2	6		•	•	^	^	^
Logical Shift Left	LSL	48	1	2	58	1	2	38	2	5	78	1	5	68	2	6		•	•	^	^	^
Logical Shift Right	LSR	44	1	2	54	1	2	34	2	5	74	1	5	64	2	6		•	•	0	^	^
Arithmetic Shift Right	ASR	47	1	2	57	1	2	37	2	5	77	1	5	67	2	6		•	•	^	^	^
Arithmetic Shift Left	ASL	48	1	2	58	1	2	38	2	5	78	1	5	68	2	6	Equal to LSL	•	•	^	^	^
Test for Negative or Zero	TST	4D	1	2	5D	1	2	3D	2	4	7D	1	4	6D	2	5	A−00 or X−00 or M−00	•	•	^	^	•

Symbols: Op = Operation
= Number of bytes
~ = Number of cycles

Table 7 Branch Instructions

Operations	Mnemonic	Addressing Modes Relative			Branch Test	Condition Code				
		OP	#	~		H	I	N	Z	C
Branch Always	BRA	20	2	3	None	•	•	•	•	•
Branch Never	BRN	21	2	3	None	•	•	•	•	•
Branch IF Higher	BHI	22	2	3	C+Z=0	•	•	•	•	•
Branch IF Lower or Same	BLS	23	2	3	C+Z=1	•	•	•	•	•
Branch IF Carry Clear	BCC	24	2	3	C=0	•	•	•	•	•
(Branch IF Higher or Same)	(BHS)	24	2	3	C=0	•	•	•	•	•
Branch IF Carry Set	BCS	25	2	3	C=1	•	•	•	•	•
(Branch IF Lower)	(BLO)	25	2	3	C=1	•	•	•	•	•
Branch IF Not Equal	BNE	26	2	3	Z=0	•	•	•	•	•
Branch IF Equal	BEQ	27	2	3	Z=1	•	•	•	•	•
Branch IF Half Carry Clear	BHCC	28	2	3	H=0	•	•	•	•	•
Branch IF Half Carry Set	BHCS	29	2	3	H=1	•	•	•	•	•
Branch IF Plus	BPL	2A	2	3	N=0	•	•	•	•	•
Branch IF Minus	BMI	2B	2	3	N=1	•	•	•	•	•
Branch IF Interrupt Mask Bit is Clear	BMC	2C	2	3	I=0	•	•	•	•	•
Branch IF Interrupt Mask Bit is Set	BMS	2D	2	3	I=1	•	•	•	•	•
Branch IF Interrupt Line is Low	BIL	2E	2	3	INT=0	•	•	•	•	•
Branch IF Interrupt Line is High	BIH	2F	2	3	INT=1	•	•	•	•	•
Branch to Subroutine	BSR	AD	2	5	—	•	•	•	•	•

Symbols: Op = Operation
= Number of bytes
~ = Number of cycles

Table 8 Bit Manipulation Instructions

Operations	Mnemonic	Addressing Modes Bit Set Clear			Bit Test and Branch			Boolean/ Arithmetic Operation	Branch Test	Condition Code				
		OP	#	~	OP	#	~			H	I	N	Z	C
Branch IF Bit n is set	BRSET n(n=0···7)	—			2·n	3	5	—	Mn=1	•	•	•	•	↕
Branch IF Bit n is clear	BRCLR n(n=0···7)				01+2·n	3	5	—	Mn=0	•	•	•	•	↕
Set Bit n	BSET n(n=0···7)	10+2·n	2	5				1→Mn	—	•	•	•	•	•
Clear Bit n	BCLR n(n=0···7)	11+2·n	2	5				0→Mn	—	•	•	•	•	•

Symbols: Op = Operation
= Number of bytes
~ = Number of cycles

Table 9 Control Instructions

Operations	Mnemonic	Addressing Modes: Implied OP	#	~	Boolean Operation	Condition Code: H	I	N	Z	C
Transfer A to X	TAX	97	1	2	A→X	●	●	●	●	●
Transfer X to A	TXA	9F	1	2	X→A	●	●	●	●	●
Set Carry Bit	SEC	99	1	1	1→C	●	●	●	●	1
Clear Carry Bit	CLC	98	1	1	0→C	●	●	●	●	0
Set Interrupt Mask Bit	SEI	9B	1	2	1→I	●	1	●	●	●
Clear Interrupt Mask Bit	CLI	9A	1	2	0→I	●	0	●	●	●
Software Interrupt	SWI	83	1	10		●	1	●	●	●
Return from Subroutine	RTS	81	1	5		●	●	●	●	●
Return from Interrupt	RTI	80	1	8		?	?	?	?	?
Reset Stack Pointer	RSP	9C	1	2	$FF→SP	●	●	●	●	●
No-Operation	NOP	9D	1	1	Advance Prog. Cntr Only	●	●	●	●	●
Decimal Adjust A	DAA	8D	1	2	Converts binary add of BCD charcters into BCD format	●	●	^	^	^*
Stop	STOP	8E	1	4		●	●	●	●	●
Wait	WAIT	8F	1	4		●	●	●	●	●

Symbols: Op = Operation
= Number of bytes
~ = Number of cycles

* Are BCD characters of upper byte 10 or more? (They are not cleared if set in advance.)

Table 10 Instruction Set (in Alphabetical Order)

Mnemonic	Addressing Modes: Implied	Immediate	Direct	Extended	Relative	Indexed (No Offset)	Indexed (8-Bit)	Indexed (16-Bit)	Bit Set/ Clear	Bit Test & Branch	Condition Code: H	I	N	Z	C
ADC		×	×	×		×	×	×			^	●	^	^	^
ADD		×	×	×		×	×	×			^	●	^	^	^
AND		×	×	×		×	×	×			●	●	^	^	●
ASL	×		×			×	×				●	●	^	^	^
ASR	×		×			×	×				●	●	^	^	^
BCC					×						●	●	●	●	●
BCLR									×		●	●	●	●	●
BCS					×						●	●	●	●	●
BEQ					×						●	●	●	●	●
BHCC					×						●	●	●	●	●
BHCS					×						●	●	●	●	●
BHI					×						●	●	●	●	●
(BHS)					×						●	●	●	●	●
BIH					×						●	●	●	●	●
BIL					×						●	●	●	●	●
BIT		×	×	×		×	×	×			●	●	^	^	●
(BLO)					×						●	●	●	●	●
BLS					×						●	●	●	●	●
BMC					×						●	●	●	●	●
BMI					×						●	●	●	●	●
BMS					×						●	●	●	●	●
BNE					×						●	●	●	●	●
BPL					×						●	●	●	●	●
BRA					×						●	●	●	●	●

(to be continued)

Condition Code Symbols:

H	Half Carry (From Bit 3)	C	Carry Borrow
I	Interrupt Mask	^	Test and Set if True, Cleared Otherwise
N	Negative (Sign Bit)	●	Not Affected
Z	Zero	?	Load CC Register From Stack

Table 10 Instruction Set (in Alphabetical Order)

Mnemonic	Addressing Modes										Condition Code				
	Implied	Immediate	Direct	Extended	Relative	Indexed (No Offset)	Indexed (8-Bit)	Indexed (16-Bit)	Bit Set Clear	Bit Test & Branch	H	I	N	Z	C
BRN					×						•	•	•	•	•
BRCLR										×	•	•	•	•	^
BRSET										×	•	•	•	•	^
BSET									×		•	•	•	•	•
BSR					×						•	•	•	•	•
CLC	×										•	•	•	•	0
CLI	×										•	0	•	•	•
CLR	×		×			×	×				•	•	0	1	•
CMP		×	×	×		×	×	×			•	•	^	^	^
COM	×		×			×	×				•	•	^	^	1
CPX		×	×	×		×	×	×			•	•	^	^	^
DAA	×										•	•	^	^	^
DEC	×		×			×	×				•	•	^	^	•
EOR		×	×	×		×	×	×			•	•	^	^	•
INC	×		×			×	×				•	•	^	^	•
JMP			×	×		×	×	×			•	•	•	•	•
JSR			×	×		×	×	×			•	•	•	•	•
LDA		×	×	×		×	×	×			•	•	^	^	•
LDX		×	×	×		×	×	×			•	•	^	^	•
LSL	×		×			×	×				•	•	^	^	^
LSR	×		×			×	×				•	•	0	^	^
NEG	×		×			×	×				•	•	^	^	^
NOP	×										•	•	•	•	•
ORA		×	×	×		×	×	×			•	•	^	^	•
ROL	×		×			×	×				•	•	^	^	^
ROR	×		×			×	×				•	•	^	^	^
RSP	×										•	•	•	•	•
RTI	×										?	?	?	?	?
RTS	×										•	•	•	•	•
SBC		×	×	×		×	×	×			•	•	^	^	^
SEC	×										•	•	•	•	1
SEI	×										•	1	•	•	•
STA			×	×		×	×	×			•	•	^	^	•
STOP	×										•	•	•	•	■
STX			×	×		×	×	×			•	•	^	^	•
SUB		×	×	×		×	×	×			•	•	^	^	^
SWI	×										•	1	•	•	■
TAX	×										•	•	•	•	•
TST	×		×			×	×				•	•	^	^	•
TXA	×										•	•	•	•	■
WAIT	×										•	•	•	•	■

Condition Code Symbols:

H	Half Carry (From Bit 3)	C	Carry Borrow
I	Interrupt Mask	^	Test and Set if True, Cleared Otherwise
N	Negative (Sign Bit)	•	Not Affected
Z	Zero	?	Load CC Register From Stack

Table 11 Operation Code Map

	Bit Manipulation		Branch	Read/Modify/Write					Control		Register/Memory						
	Test & Branch	Set/ Clear	Rel	DIR	A	X	,X1	,X0	IMP	IMP	IMM	DIR	EXT	,X2	,X1	,X0	
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	← HIGH
0	BRSET0	BSET0	BRA	NEG					RTI*	—	SUB						0
1	BRCLR0	BCLR0	BRN	—					RTS*	—	CMP						1
2	BRSET1	BSET1	BHI							—	SBC						2
3	BRCLR1	BCLR1	BLS	COM					SWI*	—	CPX						3
4	BRSET2	BSET2	BCC	LSR					—	—	AND						4
5	BRCLR2	BCLR2	BCS						—	—	BIT						5
6	BRSET3	BSET3	BNE	ROR							LDA						6
7	BRCLR3	BCLR3	BEQ	ASR					—	TAX*	→	STA				STA(+1)	7
8	BRSET4	BSET4	BHCC	LSL ASL					—	CLC	EOR						8
9	BRCLR4	BCLR4	BHCS	ROL					—	SEC	ADC						9
A	BRSET5	BSET5	BPL	DEC					—	CLI*	ORA						A
B	BRCLR5	BCLR5	BMI							SEI*	ADD						B
C	BRSET6	BSET6	BMC	INC					—	RSP*	—	JMP(−1)					C
D	BRCLR6	BCLR6	BMS	TST(−1)	TST			TST(−1)	DAA*	NOP	BSR*	JSR(+2)		JSR(+1)		JSR(+2)	D
E	BRSET7	BSET7	BIL	—					STOP*	—	LDX						E
F	BRCLR7	BCLR7	BIH	CLR					WAIT*	TXA*	—	STX				STX(+1)	F
	3/5	2/5	2/3	2/5	1/2	1/2	2/6	1/5	1/*	1/1	2/2	2/3	3/4	3/5	2/4	1/3	

LOW

(NOTES) 1. "—" is an undefined operation code.

2. The lowermost numbers in each column represent a byte count and the number of cycles required (byte count/number of cycles). The number of cycles for the mnemonics asterisked (*) is as follows:

RTI	8	TAX	2
RTS	5	RSP	2
SWI	10	TXA	2
DAA	2	BSR	5
STOP	4	CLI	2
WAIT	4	SEI	2

3. The parenthesized numbers must be added to the cycle count of the particular instruction.

- **Additional Instructions**

The following new instructions are used on the HD6305X2:

DAA Converts the contents of the accumulator into BCD code.

WAIT Causes the MPU to enter the wait mode. For this mode, see the topic, Wait Mode.

STOP Causes the MPU to enter the stop mode. For this mode, see the topic, Stop Mode.

■ OPERATION AT EACH INSTRUCTION CYCLE

The HD6305X2 employs a mechanism of the pipeline control for the instruction fetch and the subsequent instruction fetch is performed during the current instruction being executed.

Table 12 provides the information about the relationship among each data on the Address Bus, Data Bus and R/$\overline{W}$ status in cycle-by-cycle basis during the execution of each instruction.

Table 12 Cycle-by-Cycle Operation

Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/$\overline{W}$	Data Bus
IMMEDIATE					
ADC, ADD, AND, BIT, CMP, CPX, EOR, LDA, LDX, ORA, SBC, SUB	2	1	Op Code Address +1	1	Operand Data
		2	Op Code Address +2	1	Next Op Code
DIRECT					
ADC, ADD, AND, BIT, CMP, CPX, EOR, LDA, LDX, ORA, SBC, SUB	3	1	Op Code Address +1	1	Address of Operand
		2	Address of Operand	1	Operand Data
		3	Op Code Address +2	1	Next Op Code

(to be continued)

Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/$\overline{W}$	Data Bus
STA, STX	3	1	Op Code Address +1	1	Address of Operand
		2	Address of Operand	0	Data from Acc. / Data from Ix.
		3	Op Code Address +1	1	Next Op Code
JMP	2	1	Op Code Address +1	1	Jump Address
		2	Jump Address	1	Next Op Code
JSR	5	1	Op Code Address +1	1	Jump Address (LSB)
		2	1FFF	1	Irrelevant Data
		3	Stack Pointer	0	Return Address (LSB)
		4	Stack Pointer -1	0	Return Address (MSB)
		5	Jump Address	1	First Subroutine Op Code
ASR, CLR, COM,	5	1	Op Code Address +1	1	Address of Operand
DEC, INC, LSL,		2	Address of Operand	1	Operand Data
LSR, NEG, ROL,		3	1FFF	1	Irrelevant Data
ROR		4	Address of Operand	0	New Operand Data
		5	Op Code Address +2	1	Next Op Code
TST	4	1	Op Code Address +1	1	Address of Operand
		2	Address of Operand	1	Operand Data
		3	1FFF	1	Irrelevant Data
		4	Op Code Address +2	1	Next Op Code
EXTENDED					
ADC, ADD, AND,	4	1	Op Code Address +1	1	Address of Operand (MSB)
BIT, CMP, CPX,		2	Op Code Address +2	1	Address of Operand (LSB)
EOR, LDA, LDX,		3	Address of Operand	1	Operand Data
ORA, SBC, SUB		4	Op Code Address +3	1	Next Op Code
STA, STX	4	1	Op Code Address +1	1	Address of Operand (MSB)
		2	Op Code Address +2	1	Address of Operand (LSB)
		3	Address of Operand	0	Data from Acc. / Data from Ix.
		4	Op Code Address +3	1	Next Op Code
JMP	3	1	Op Code Address +1	1	Jump Address (MSB)
		2	Op Code Address +2	1	Jump Address (LSB)
		3	Jump Address	1	Next Op Code
JSR	6	1	Op Code Address +1	1	Jump Address (MSB)
		2	Op Code Address +2	1	Jump Address (LSB)
		3	1FFF	1	Irrelevant Data
		4	Stack Pointer	0	Return Address (LSB)
		5	Stack Pointer -1	0	Return Address (MSB)
		6	Jump Address	1	First Subroutine Op Code
INDEXED (No offset)					
ADC, ADD, AND,	3	1	Op Code Address +1	1	Next Op Code
BIT, CMP, CPX,		2	Ix	1	Operand Data
EOR, LDA, LDX,		3	Op Code Address +1	1	Next Op Code
ORA, SBC, SUB					
STA, STX	4	1	Op Code Address +1	1	Next Op Code
		2	1FFF	1	Irrelevant Data
		3	Ix	0	Data from Acc. / Data from Ix.
		4	Op Code Address +1	1	Next Op Code
JMP	2	1	Op Code Address +1	1	Next Op Code
		2	Ix	1	First Op Code of Jump Routine

(to be continued)

Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/W	Data Bus
JSR	5	1	Op Code Address +1	1	Next Op Code
		2	1FFF	1	Irrelevant Data
		3	Stack Pointer	0	Return Address (LSB)
		4	Stack Pointer –1	0	Return Address (MSB)
		5	Ix	1	First Subroutine Op Code
ASR, CLR, COM,	5	1	Op Code Address +1	1	Next Op Code
DEC, INC, LSL,		2	Ix	1	Operand Data
LSR, NEG, ROL,		3	1FFF	1	Irrelevant Data
ROR		4	Ix	0	New Operand Data
		5	Op Code Address +1	1	Next Op Code
TST	4	1	Op Code Address +1	1	Next Op Code
		2	Ix	1	Operand Data
		3	1FFF	1	Irrelevant Data
		4	Op Code Address +1	1	Next Op Code
INDEXED (8-bit offset)					
ADC, ADD, AND,	4	1	Op Code Address +1	1	Offset
BIT, CMP, CPX,		2	1FFF	1	Irrelevant Data
EOR, LDA, LDX,		3	Ix + Offset	1	Operand Data
ORA, SBC, SUB		4	Op Code Address +2	1	Next Op Code
STA, STX	4	1	Op Code Address +1	1	Offset
		2	1FFF	1	Irrelevant Data
		3	Ix + Offset	0	{ Data from Acc. / Data from Ix.
		4	Op Code Address +2	1	Next Op Code
JMP	3	1	Op Code Address +1	1	Offset
		2	1FFF	1	Irrelevant Data
		3	Ix + Offset	1	First Op Code of Jump Routine
JSR	5	1	Op Code Address +1	1	Offset
		2	1FFF	1	Irrelevant Data
		3	Stack Pointer	0	Return Address (LSB)
		4	Stack Pointer –1	0	Return Address (MSB)
		5	Ix + Offset	1	First Subroutine Op Code
ASR, CLR, COM,	6	1	Op Code Address +1	1	Offset
DEC, INC, LSL,		2	1FFF	1	Irrelevant Data
LSR, NEG, ROL,		3	Ix + Offset	1	Operand Data
ROR		4	1FFF	1	Irrelevant Data
		5	Ix + Offset	0	New Operand Data
		6	Op Code Address +1	1	Next Op Code
TST	5	1	Op Code Address +1	1	Offset
		2	1FFF	1	Irrelevant Data
		3	Ix + Offset	1	Operand Data
		4	1FFF	1	Irrelevant Data
		5	Op Code Address +2	1	Next Op Code
INDEXED (16-bit offset)					
ADC, ADD, AND,	5	1	Op Code Address +1	1	Offset (MSB)
BIT, CMP, CPX,		2	Op Code Address +2	1	Offset (LSB)
EOR, LDA, LDX,		3	1FFF	1	Irrelevant Data
ORA, SBC, SUB		4	Ix + Offset	1	Operand Data
		5	Op Code Address +1	1	Next Op Code

(to be continued)

Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/W	Data Bus
STA, STX	5	1	Op Code Address +1	1	Offset (MSB)
		2	Op Code Address +2	1	Offset (LSB)
		3	1FFF	1	Irrelevant Data
		4	Ix + Offset	0	Data from Acc. / Data from Ix.
		5	Op Code Address +3	1	Next Op Code
JMP	4	1	Op Code Address +1	1	Offset (MSB)
		2	Op Code Address +2	1	Offset (LSB)
		3	1FFF	1	Irrelevant Data
		4	Ix + Offset	1	First Op Code of Jump Routine
JSR	6	1	Op Code Address +1	1	Offset (MSB)
		2	Op Code Address +2	1	Offset (LSB)
		3	1FFF	1	Irrelevant Data
		4	Stack Pointer	0	Return Address (LSB)
		5	Stack Pointer -1	0	Return Address (MSB)
		6	Ix + Offset	1	First Subroutine Op Code
IMPLIED					
ASR, CLR, COM, DEC, INC, LSL, LSR, NEG, ROL, ROR, TST	2	1	Op Code Address +1	1	Next Op Code
		2	Op Code Address +1	1	Next Op Code
CLC, NOP, SEC	1	1	Op Code Address +1	1	Next Op Code
RSP, TAX, TXA	2	1	Op Code Address +1	1	Next Op Code
		2	Op Code Address +1	1	Next Op Code
CLI, SEI	2	1	Op Code Address +1	1	Next Op Code
		2	1FFF	1	Irrelevant Data
DAA	2	1	Op Code Address +1	1	Next Op Code
		2	Op Code Address +1	1	Next Op Code
STOP, WAIT	4	1	Op Code Address +1	1	Next Op Code
		2	1FFF	1	Irrelevant Data
		3	1FFF	1	Irrelevant Data
		4	Op Code Address +1	1	Next Op Code
RTI	8	1	Op Code Address +1	1	Next Op Code
		2	1FFF	1	Irrelevant Data
		3	Stack Pointer	1	CC
		4	Stack Pointer +1	1	Acc.
		5	Stack Pointer +2	1	Ix.
		6	Stack Pointer +3	1	Return Address (MSB)
		7	Stack Pointer +4	1	Return Address (LSB)
		8	Return Address	1	First Op Code of Return Routine
RTS	5	1	Op Code Address +1	1	Next Op Code
		2	1FFF	1	Irrelevant Data
		3	Stack Pointer	1	Return Address (MSB)
		4	Stack Pointer +1	1	Return Address (LSB)
		5	Return Address	1	First Op Code of Return Routine
SWI	10	1	Op Code Address +1	1	Next Op Code
		2	1FFF	1	Irrelevant Data
		3	Stack Pointer	0	Return Address (LSB)
		4	Stack Pointer-1	0	Return Address (MSB)
		5	Stack Pointer-2	0	Ix.
		6	Stack Pointer-3	0	Acc.
		7	Stack Pointer-4	0	CC
		8	Vector Address 1FFC	1	Address of SWI Routine (MSB)
		9	Vector Address 1FFD	1	Address of SWI Routine (LSB)
		10	Address of SWI Routine	1	First Op Code of SWI Routine

(to be continued)

Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/$\overline{W}$	Data Bus
RELATIVE					
BCC, BCS, BEQ, BHCC, BHCS, BHI, BIH, BIL, BLS, BMC, BMI, BMS, BNE, BPL, BRA, BRN	3	1	Op Code Address +1	1	Next Op Code
		2	1FFF	1	Irrelevant Data
		3	Branch Address .....Test = "1" Op Code Address +1 .... Test = "0"	1	First Op Code of Branch Routine Next Op Code
BSR	5	1	Op Code Address +1	1	Offset
		2	1FFF	1	Irrelevant Data
		3	Stack Pointer	0	Return Address (LSB)
		4	Stack Pointer–1	0	Return Address (MSB)
		5	Branch Address	1	First Op Code of Subroutine
BIT TEST AND BRANCH					
BRCLR, BRSET	5	1	Op Code Address +1	1	Address of Operand
		2	Address of Operand	1	Operand Data
		3	Op Code Address +2	1	Offset
		4	1FFF	1	Irrelevant Data
		5	Branch Address .....Test ="1" Op Code Address +3 .....Test = "0"	1	First Op Code of Branch Address Next Op Code
BIT SET/CLEAR					
BCLR, BSET	5	1	Op Code Address +1	1	Address of Operand
		2	Address of Operand	1	Operand Data
		3	1FFF	1	Irrelevant Data
		4	Address of Operand	0	New Operand Data
		5	Op Code Address +1	1	Next Op Code

HD6305Y2, HD63A05Y2, HD63B05Y2
CMOS MPU (Micro Processing Unit)

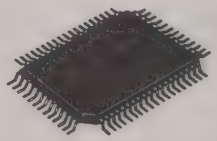
The HD6305Y2 is a CMOS 8-bit micro processing unit. A CPU, a clock generator, a 256 byte RAM, I/O terminals, two timers and a serial communication interface (SCI) are built in the HD6305Y2.

The HD6305Y2 has the same functions as the HD6305Y0's except for the number of I/O terminals. Its memory space is expandable to 16k bytes externally.

HD6305Y2P, HD63A05Y2P, HD63B05Y2P

(DP-64S)

HD6305Y2F, HD63A05Y2F, HD63B05Y2F



(FP-64)

■ HARDWARE FEATURES

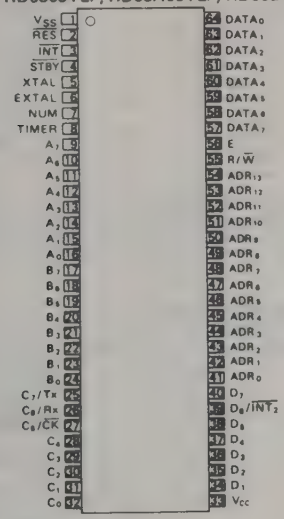
- 8-bit based MPU
- 256 bytes of RAM
- A total of 31 terminals, including 24 I/O's, 7 inputs
- Two timers
  - 8-bit timer with a 7-bit prescaler (programmable prescaler; event counter)
  - 15-bit timer (commonly used with the SCI clock divider)
- On-chip serial interface circuit (synchronized with clock)
- Six interrupts (two external, two timer, one serial and one software)
- Low power dissipation modes
  - Wait . . . . In this mode, the clock oscillator is on and the CPU halts but the timer/serial/interrupt function is operatable.
  - Stop . . . . In this mode, the clock stops but the RAM data, I/O status and registers are held.
  - Standby. . In this mode, the clock stops, the RAM data is held, and the other internal condition is reset.
- Minimum instruction cycle time
  - HD6305Y2 . . . . . 1 μs (f = 1 MHz)
  - HD63A05Y2. . . . . 0.67 μs (f = 1.5 MHz)
  - HD63B05Y2 . . . . 0.5 μs (f = 2 MHz)
- Wide operating range
  V_{CC} = 3 to 6V (f = 0.1 to 0.5 MHz)
  - HD6305Y2 . . . . . f = 0.1 to 1 MHz (V_{CC} = 5V ± 10%)
  - HD63A05Y2 . . . . f = 0.1 to 1.5 MHz (V_{CC} = 5 V ± 10%)
  - HD63B05Y2 . . . . f = 0.1 to 2 MHz (V_{CC} = 5 V ± 10%)
- System development fully supported by an evaluation kit

■ SOFTWARE FEATURES

- Similar to HD6800
- Byte efficient instruction set
- Powerful bit manipulation instructions (Bit Set, Bit Clear, and Bit Test and Branch usable for 192 byte RAM bits within page 0 and all I/O terminals)
- A variety of interrupt operations
- Index addressing mode useful for table processing
- A variety of conditional branch instructions
- Ten powerful addressing modes
- All addressing modes adaptable to RAM, and I/O instructions
- Three new instructions, STOP, WAIT and DAA, added to the HD6805 family instruction set
- Instructions that are upward compatible with those of Motorola's MC6805P2 and MC146805G2

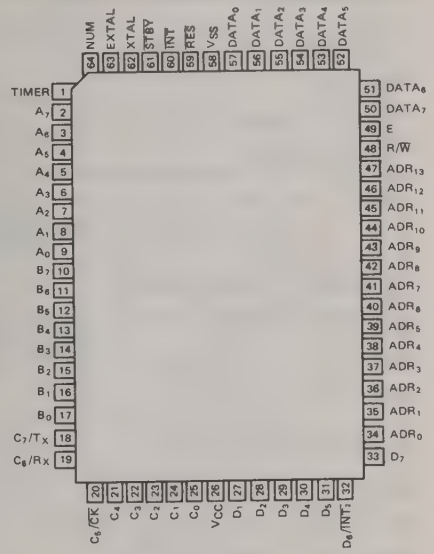
■ PIN ARRANGEMENT

● HD6305Y2P, HD63A05Y2P, HD63B05Y2P



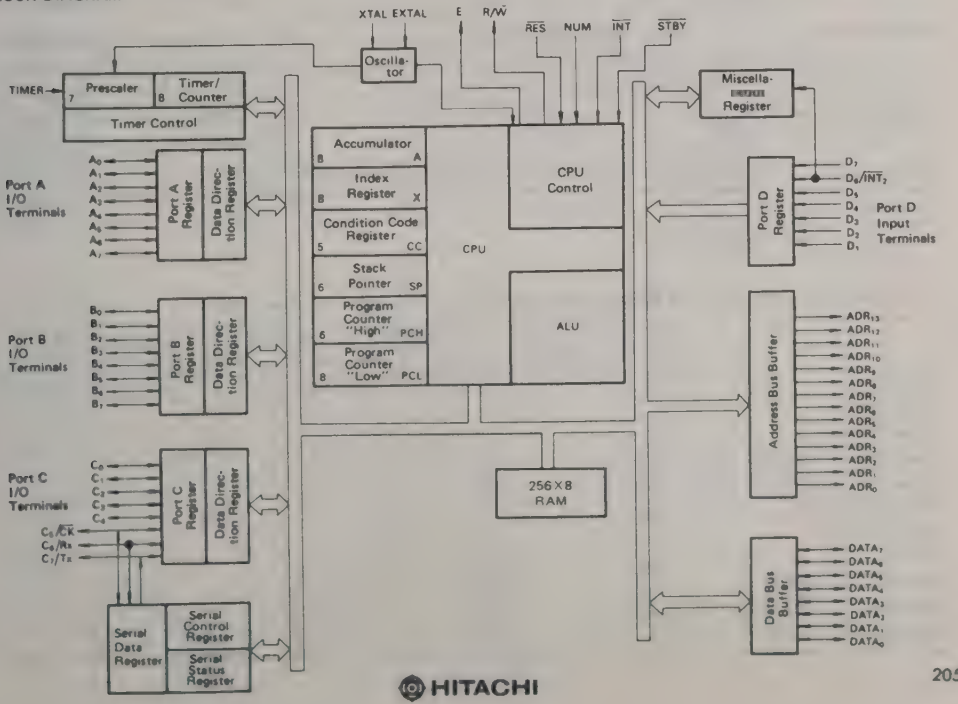
(Top View)

● HD6305Y2F, HD63A05Y2F, HD63B05Y2F



(Top View)

■ BLOCK DIAGRAM



■ ABSOLUTE MAXIMUM RATINGS

Item	Symbol	Value	Unit
Supply Voltage	V_{CC}	−0.3 ~ +7.0	V
Input Voltage	V_{in}	−0.3 ~ V_{CC} + 0.3	V
Operating Temperature	T_{opr}	0 ~ +70	°C
Storage Temperature	T_{stg}	−55 ~ +150	°C

[NOTE] These products have a protection circuit in their input terminals against high electrostatic voltage or high electric fields. Notwithstanding, be careful not to apply any voltage higher than the absolute maximum rating to these high input impedance circuits. To assure normal operation, we recommended V_{In}, V_{out}; $V_{SS} \leqq (V_{In}$ or $V_{out}) \leqq V_{CC}$.

■ ELECTRICAL CHARACTERISTICS

● DC CHARACTERISTICS (V_{CC} = 5.0V±10%, V_{SS} = 0V, Ta = 0 ~ +70°C, unless otherwise noted.)

Item		Symbol	Test Condition	min	typ	max	Unit
Input "High" Voltage	$\overline{RES}$, $\overline{STBY}$	V_{IH}		V_{CC}−0.5	—	V_{CC}+0.3	V
	EXTAL			V_{CC}x0.7	—	V_{CC}+0.3	
	Other Inputs			2.0	—	V_{CC}+0.3	
Input "Low" Voltage	All Inputs	V_{IL}		−0.3	—	0.8	V
Output "High" Voltage	All Outputs	V_{OH}	I_{OH} = −200μA	2.4	—	—	V
			I_{OH} = −10μA	V_{CC}−0.7	—	—	
Output "Low" Voltage	All Outputs	V_{OL}	I_{OL} = 1.6mA	—	—	0.55	V
Input Leakage Current	TIMER, $\overline{INT}$, $D_1 \sim D_7$, $\overline{STBY}$	$\|I_{IL}\|$	Vin = 0.5 ~ V_{CC}−0.5	—	—	1.0	μA
Three-state Current	$A_0 \sim A_7$, $B_0 \sim B_7$, $C_0 \sim C_7$, $ADR_0 \sim ADR_{13}$*, $DATA_0 \sim DATA_7$, E*, R/$\overline{W}$*	$\|I_{TSI}\|$		—	—	1.0	μA
Current Dissipation**	Operating	I_{CC}	f = 1MHz***	—	5	10	mA
	Wait			—	2	5	mA
	Stop			—	2	10	μA
	Standby			—	2	10	μA
Input Capacitance	All Terminals	Cin	f = 1MHz, Vin = 0V	—	—	12	pF

* Only at standby
** V_{IH} min = V_{CC}−1.0V, V_{IL} max = 0.8V
*** The value at f = x MHz is given by using.
I_{CC} (f = x MHz) = I_{CC} (f = 1MHz) x x

● AC CHARACTERISTICS (V_{CC} = 5.0V±10%, V_{SS} = 0V, Ta = 0 ~ +70°C, unless otherwise noted.)

Item	Symbol	Test Condition	HD6305Y2			HD63A05Y2			HD63B05Y2			Unit
			min	typ	max	min	typ	max	min	typ	max	
Cycle Time	t_{cyc}	Fig. 1	1	—	10	0.666	—	10	0.5	—	10	μs
Enable Rise Time	t_{Er}		—	—	20	—	—	20	—	—	20	ns
Enable Fall Time	t_{Ef}		—	—	20	—	—	20	—	—	20	ns
Enable Pulse Width("High" Level)	PW_{EH}		450	—	—	300	—	—	220	—	—	ns
Enable Pulse Width("Low" Level)	PW_{EL}		450	—	—	300	—	—	220	—	—	ns
Address Delay Time	t_{AD}		—	—	250	—	—	190	—	—	180	ns
Address Hold Time	t_{AH}		40	—	—	30	—	—	20	—	—	ns
Data Delay Time	t_{DW}		—	—	200	—	—	160	—	—	120	ns
Data Hold Time (Write)	t_{HW}		40	—	—	30	—	—	20	—	—	ns
Data Set-up Time (Read)	t_{DSR}		80	—	—	60	—	—	50	—	—	ns
Data Hold Time (Read)	t_{HR}		0	—	—	0	—	—	0	—	—	ns

- **PORT TIMING** (V_{CC} = 5.0V±10%, V_{SS} = 0V, Ta = 0 ~ +70°C, unless otherwise noted.)

Item	Symbol	Test Condition	HD6305Y2			HD63A05Y2			HD63B05Y2			Unit
			min	typ	max	min	typ	max	min	typ	max	
Port Data Set-up Time (Port A, B, C, D)	t_{PDS}	Fig. 2	200	–	–	200	–	–	200	–	–	ns
Port Data Hold Time (Port A, B, C, D)	t_{PDH}		200	–	–	200	–	–	200	–	–	ns
Port Data Delay Time (Port A, B, C)	t_{PDW}	Fig. 3	–	–	300	–	–	300	–	–	300	ns

- **CONTROL SIGNAL TIMING** (V_{CC} = 5.0V±10%, V_{SS} = 0V, Ta = 0 ~ +70°C, unless otherwise noted.)

Item	Symbol	Test Condition	HD6305Y2			HD63A05Y2			HD63B05Y2			Unit
			min	typ	max	min	typ	max	min	typ	max	
$\overline{INT}$ Pulse Width	t_{IWL}		t_{cyc} +250	–	–	t_{cyc} +200	–	–	t_{cyc} +200	–	–	ns
$\overline{INT_2}$ Pulse Width	t_{IWL2}		t_{cyc} +250	–	–	t_{cyc} +200	–	–	t_{cyc} +200	–	–	ns
$\overline{RES}$ Pulse Width	t_{RWL}		5	–	–	5	–	–	5	–	–	t_{cyc}
Control Set-up Time	t_{CS}	Fig. 5	250	–	–	250	–	–	250	–	–	ns
Timer Pulse Width	t_{TWL}		t_{cyc} +250	–	–	t_{cyc} +200	–	–	t_{cyc} +200	–	–	ns
Oscillation Start Time (Crystal)	t_{OSC}	Fig.5, Fig.20*	–	–	20	–	–	20	–	–	20	ms
Reset Delay Time	t_{RHL}	Fig. 19	80	–	–	80	–	–	80	–	–	ms

* C_L = 22pF ±20%, R_s = 60Ω max.

- **SCI TIMING** (V_{CC} = 5.0V±10%, V_{SS} = 0V, Ta = 0 ~ +70°C, unless otherwise noted.)

Item	Symbol	Test Condition	HD6305Y2			HD63A05Y2			HD63B05Y2			Unit
			min	typ	max	min	typ	max	min	typ	max	
Clock Cycle	t_{Scyc}	Fig. 6, Fig. 7	1	–	32768	0.67	–	21845	0.5	–	16384	μs
Data Output Delay Time	t_{TXD}		–	–	250	–	–	250	–	–	250	ns
Data Set-up Time	t_{SRX}		200	–	–	200	–	–	200	–	–	ns
Data Hold Time	t_{HRX}		100	–	–	100	–	–	100	–	–	ns

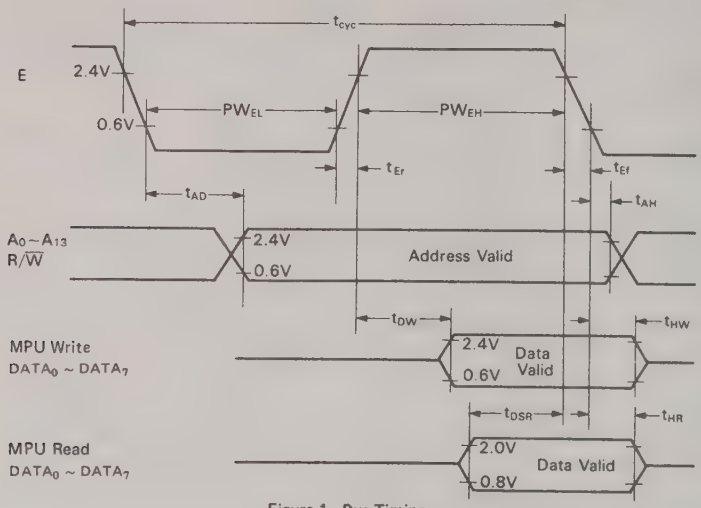


Figure 1 Bus Timing

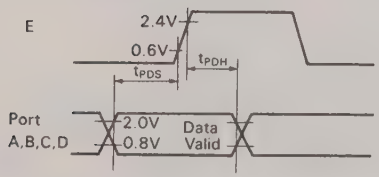


Figure 2 Port Data Set-up and Hold Times (MPU Read)

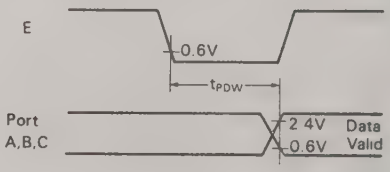


Figure 3 Port Data Delay Time (MPU Write)

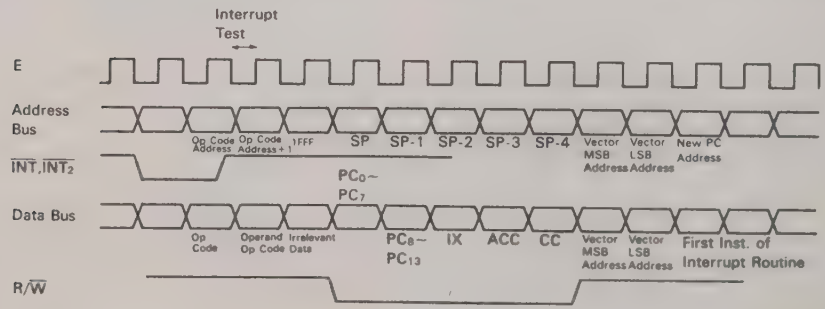


Figure 4 Interrupt Sequence

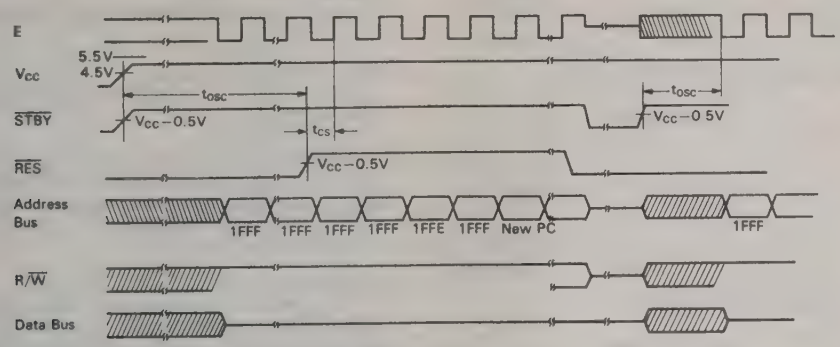


Figure5 Reset Timing

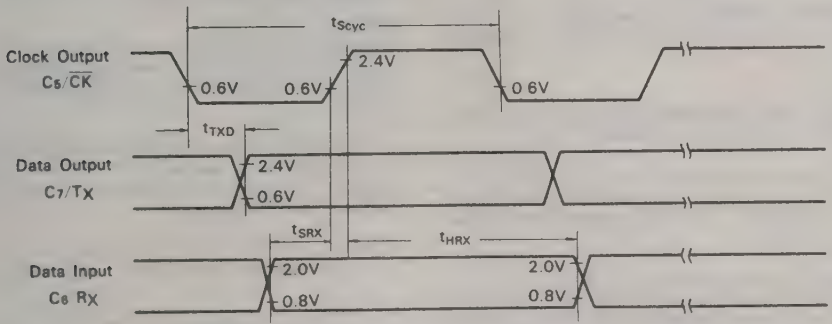


Figure6 SCI Timing (Internal Clock)

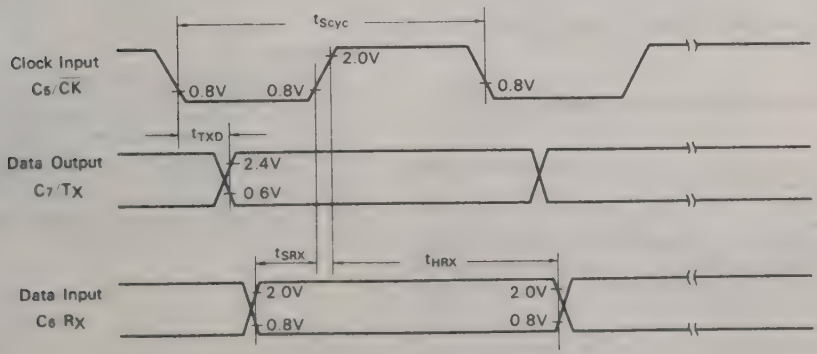
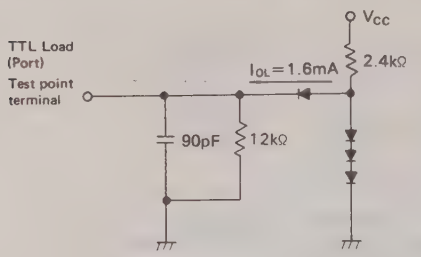


Figure7 SCI Timing(External Clock)



[NOTES] 1. The load capacitance includes stary capacitance caused by the probe, etc.
2. All diodes are 1S2074 (H).

Figure 8 Test Load

■ DESCRIPTION OF TERMINAL FUNCTIONS

The input and output signals of the MPU are described here.

• V_{CC}, V_{SS}

Voltage is applied to the MPU through these two terminals. V_{CC} is 5.0V ± 10%, while V_{SS} is grounded.

• $\overline{INT}$, $\overline{INT_2}$

External interrupt request inputs to the MPU. For details, refer to "INTERRUPT". The $\overline{INT_2}$ terminal is also used as the port D_6 terminal.

• XTAL, EXTAL

These terminals provide input to the on-chip clock circuit. A crystal oscillator (AT cut, 2.0 to 8.0 MHz) or ceramic filter is connected to the terminal. Refer to "INTERNAL OSCILLATOR" for using these input terminals.

• TIMER

This is an input terminal for event counter. Refer to "TIMER" for details.

• $\overline{RES}$

Used to reset the MPU. Refer to "RESET" for details.

• NUM

This terminal is not for user application. This terminal should be connected to V_{SS}.

• Enable (E)

This output terminal supplies E clock. Output is a single-phase, TTL compatible and 1/4 crystal oscillation frequency or 1/4 external clock frequency. It can drive one TTL load and a 90 pF condenser.

• Read/Write (R/$\overline{W}$)

This TTL compatible output signal indicates to peripheral and memory devices whether MPU is in Read ("High"), or in Write ("Low"). The normal standby state is Read ("High"). Its output can drive one TTL load and a 90pF condenser.

• Data Bus ($DATA_0 \sim DATA_7$)

This TTL compatible three-state buffer can drive one TTL load and 90pF.

• Address Bus ($ADR_0 \sim ADR_{13}$)

Each terminal is TTL compatible and can drive one TTL load and 90pF.

• Input/Output Terminals ($A_0 \sim A_7$, $B_0 \sim B_7$, $C_0 \sim C_7$)

These 24 terminals consist of three 8-bit I/O ports (A, B, C). Each of them can be used as an input or output terminal on a bit through program control of the data direction register. For details, refer to "I/O PORTS."

• Input Terminals ($D_1 \sim D_7$)

These seven input-only terminals are TTL or CMOS compatible. Of the port D's, D_6 is also used as $\overline{INT_2}$. If D_6 is used as a port, the $\overline{INT_2}$ interrupt mask bit of the miscellaneous register must be set to "1" to prevent an $\overline{INT_2}$ interrupt from being accidentally accepted.

• $\overline{STBY}$

This terminal is used to place the MPU into the standby mode. With $\overline{STBY}$ at "Low" level, the oscillation stops and the internal condition is reset. For details, refer to "Stand-by Mode."

The terminals described in the following are I/O pins for serial communication interface (SCI). They are also used as ports C_5, C_6 and C_7. For details, refer to "SERIAL COMMUNICATION INTERFACE."

• $\overline{CK}$ (C_5)

Used to input or output clocks for serial operation.

• Rx (C_6)

Used to receive serial data.

• Tx (C_7)

Used to transmit serial data.

■ MEMORY MAP

The memory map of the MPU is shown in Fig. 9. During interrupt processing, the contents of the CPU registers are saved into the stack in the sequence shown in Fig. 10. This saving begins with the lower byte (PCL) of the program counter. Then the value of the stack pointer is decremented and the higher byte (PCH) of the program counter, index register (X), accumulator (A) and condition code register (CC) are stacked in that order. In a subroutine call, only the contents of the program counter (PCH and PCL) are stacked.

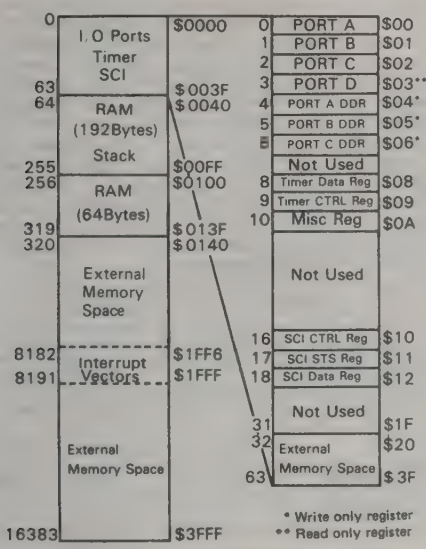


Figure 9 Memory Map of MPU

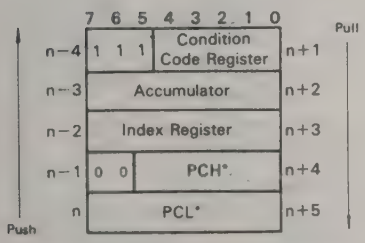


Figure 10 Sequence of Interrupt Stacking

■REGISTERS

There are five registers which the programmer can operate.

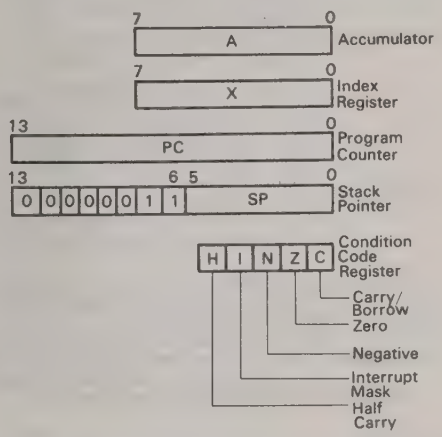


Figure 11 Programming Model

• Accumulator (A)

This accumulator is an ordinary 8-bit register which holds operands or the result of arithmetic operation or data processing.

• Index Register (X)

The index register is an 8-bit register, and is used for index addressing mode. Each of the addresses contained in the register consists of 8 bits which, combined with an offset value, provides an effective address.

In the case of a read/modify/write instruction, the index register can be used like an accumulator to hold operation data or the result of operation.

If not used in the index addressing mode, the register can be used to store data temporarily.

• Program Counter (PC)

The program counter is a 14-bit register that contains the address of the next instruction to be executed.

• Stack Pointer (SP)

The stack pointer is a 14-bit register that indicates the address of the next stacking space. Just after reset, the stack pointer is set at address $00FF. It is decremented when data is pushed, and incremented when pulled. The upper 8 bits of the stack pointer are fixed to 00000011. During the MPU being reset or during a reset stack pointer (RSP) instruction, the pointer is set to address $00FF. Since a subroutine or interrupt can use space up to address $00C1 for stacking, the subroutine can be used up to 31 levels and the interrupt up to 12 levels.

• Condition Code Register (CC)

The condition code register is a 5-bit register, each bit indicating the result of the instruction just executed. The bits can be individually tested by conditional branch instruc-

tions. The CC bits are as follows:

Half Carry (H): Used to indicate that a carry occurred between bits 3 and 4 during an arithmetic operation (ADD, ADC).

Interrupt (I): Setting this bit causes all interrupts, except a software interrupt, to be masked. If an interrupt occurs with the bit I set, it is latched. It will be processed the instant the interrupt mask bit is reset. (More specifically, it will enter the interrupt processing routine after the instruction following the CLI has been executed.)

Negative (N): Used to indicate that the result of the most recent arithmetic operation, logical operation or data processing is negative (bit 7 is logic "1").

Zero (Z): Used to indicate that the result of the most recent arithmetic operation, logical operation or data processing is zero.

Carry/Borrow (C): Represents a carry or borrow that occurred in the most recent arithmetic operation. This bit is also affected by the Bit Test and Branch instruction and a Rotate instruction.

■ INTERRUPT

There are six different types of interrupt: external interrupts ($\overline{INT}$, $\overline{INT_2}$), internal timer interrupts (TIMER, $TIMER_2$), serial interrupt (SCI) and interrupt by an instruction (SWI).

Of these six interrupts, the $\overline{INT_2}$ and TIMER or the SCI and $TIMER_2$ generate the same vector address, respectively.

When an interrupt occurs, the program in progress stops and the then CPU status is saved onto the stack. And then, the interrupt mask bit (I) of the condition code register is set and the start address of the interrupt processing routine is obtained from a particular interrupt vector address. Then the interrupt routine starts from the start address. System can exit from the interrupt routine by an RTI instruction. When this instruction is executed, the CPU status before the interrupt (saved onto the stack) is pulled and the CPU restarts the sequence with the instruction next to the one at which the interrupt occurred. Table 1 lists the priority of interrupts and their vector addresses.

Table 1 Priority of Interrupts

Interrupt	Priority	Vector Address
$\overline{RES}$	1	$1FFE, $1FFF
SWI	2	$1FFC, $1FFD
$\overline{INT}$	3	$1FFA, $1FFB
TIMER/$\overline{INT_2}$	4	$1FF8, $1FF9
SCI/$TIMER_2$	5	$1FF6, $1FF7

A flowchart of the interrupt sequence is shown in Fig. 12. A block diagram of the interrupt request source is shown in Fig. 13.

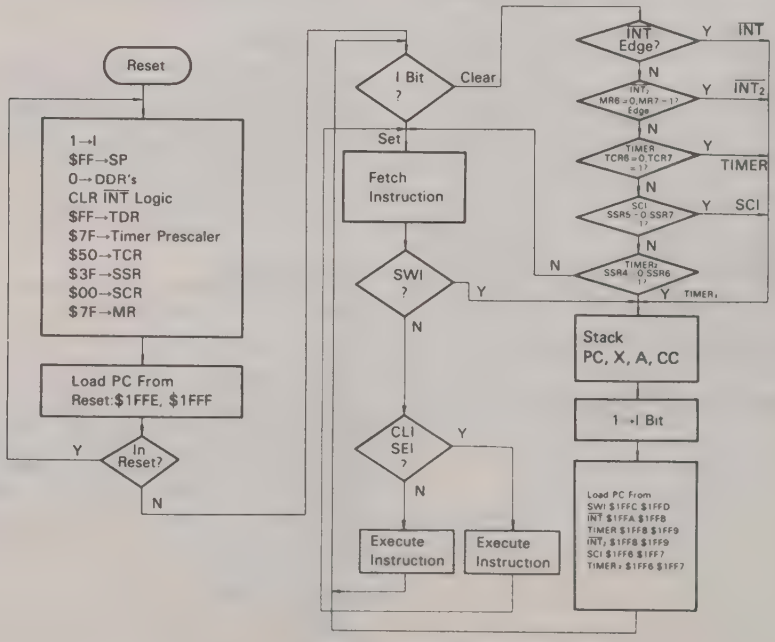


Figure 12 Interrupt Flow Chart

In the block diagram, both the external interrupts $\overline{INT}$ and $\overline{INT_2}$ are edge trigger inputs. At the falling edge of each input, an interrupt request is generated and latched. The $\overline{INT}$ interrupt request is automatically cleared if jumping is made to the $\overline{INT}$ processing routine. Meanwhile, the $\overline{INT_2}$ request is cleared if "0" is written in bit 7 of the miscellaneous register.

For the external interrupts ($\overline{INT}$, $\overline{INT_2}$), internal timer interrupts (TIMER, $TIMER_2$) and serial interrupt (SCI), each interrupt request is held, but not processed, if the I bit of the condition code register is set. Immediately after the I bit is cleared, the corresponding interrupt processing starts according to the priority.

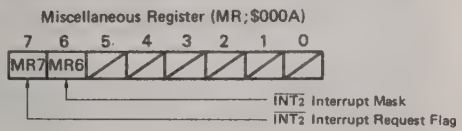
The $\overline{INT_2}$ interrupt can be masked by setting bit 6 of the miscellaneous register; the TIMER interrupt by setting bit 6 of the timer control register; the SCI interrupt by setting bit 5 of the serial status register; and the $TIMER_2$ interrupt by setting bit 4 of the serial status register.

The status of the $\overline{INT}$ terminal can be tested by a BIL or BIH instruction. The $\overline{INT}$ falling edge detector circuit and its latching circuit are independent of testing by these instructions. This is also true with the status of the $\overline{INT_2}$ terminal.

•Miscellaneous Register (MR; $000A)

The interrupt vector address for the external interrupt $\overline{INT_2}$ is the same as that for the TIMER interrupt, as shown in Table 1. For this reason, a special register called the miscellaneous register (MR; $000A) is available to control the $\overline{INT_2}$ interrupts.

Bit 7 of this register is the $\overline{INT_2}$ interrupt request flag. When the falling edge is detected at the $\overline{INT_2}$ terminal, "1" is set in bit 7. Then the software in the interrupt routine (vector addresses: $1FF8, $1FF9) checks bit 7 to see if it is $\overline{INT_2}$ interrupt. Bit 7 can be reset by software.



Bit 6 is the $\overline{INT_2}$ interrupt mask bit. If this bit is set to "1", then the $\overline{INT_2}$ interrupt is disabled. Both read and write are possible with bit 7 but "1" cannot be written in this bit by software. This means that an interrupt request by software is impossible.

When reset, bit 7 is cleared to "0" and bit 6 is set to "1".

■TIMER

Figure 14 shows a MPU timer block diagram. The timer data register is loaded by software and, upon receipt of a clock input, begins to count down. When the timer data

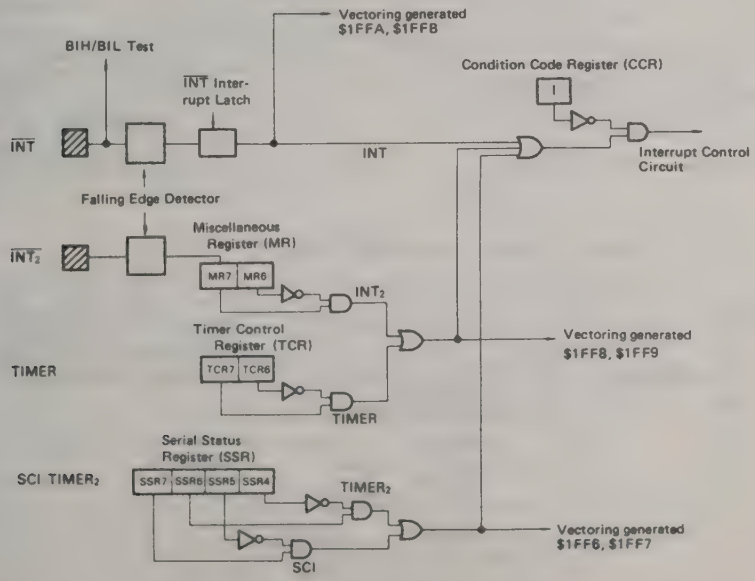


Figure 13 Interrupt Request Generation Circuitry

register (TDR) becomes "0", the timer interrupt request bit (bit 7) in the timer control register is set. In response to the interrupt request, the CPU saves its status into the stack and fetches timer interrupt routine address from addresses $1FF8 and $1FF9 and execute the interrupt routine. The timer interrupt can be masked by setting the timer interrupt mask bit (bit 6) in the timer control register. The mask bit (I) in the condition code register can also mask the timer interrupt.

The source clock to the timer can be either an external signal from the timer input terminal or the internal E signal (the oscillator clock divided by 4). If the E signal is used as the source, the clock input can be gated by the input to the timer input terminal.

Once the timer count has reached "0", it starts counting down with "$FF". The count can be monitored whenever desired by reading the timer data register. This permits the program to know the length of time having passed after the occurrence of a timer interrupt, without disturbing the contents of the counter.

When the MPU is reset, both the prescaler and counter are initialized to logic "1". The timer interrupt request bit (bit 7) then is cleared and the timer interrupt mask bit (bit 6) is set.

To clear the timer interrupt request bit (bit 7), it is necessary to write "0" in that bit.

TCR7	Timer interrupt request
0	Absent
1	Present

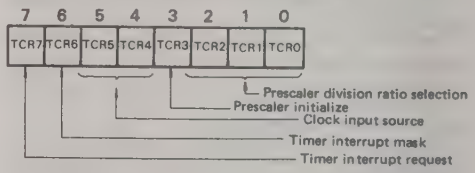
TCR6	Timer interrupt mask
0	Enabled
1	Disabled

• Timer Control Register (TCR; $0009)

Selection of a clock source, selection of a prescaler frequency division ratio, and a timer interrupt can be controlled by the timer control register (TCR; $0009).

For the selection of a clock source, any one of the four modes (see Table 2) can be selected by bits 5 and 4 of the timer control register (TCR).

Timer Control Register (TCR; $0009)


7 6 5 4 3 2 1 0
TCR7 TCR6 TCR5 TCR4 TCR3 TCR2 TCR1 TCR0
Prescaler division ratio selection
Prescaler initialize
Clock input source
Timer interrupt mask
Timer interrupt request

After reset, the TCR is initialized to "E under timer terminal control" (bit 5 = 0, bit 4 = 1). If the timer terminal is "1", the counter starts counting down with "$FF" immediately after reset.

When "1" is written in bit 3, the prescaler is initialized. This bit always shows "0" when read.

Table 2 Clock Source Selection

TCR		Clock input source
Bit 5	Bit 4	
0	0	Internal clock E
0	1	E under timer terminal control
1	0	No clock input (counting stopped)
1	1	Event input from timer terminal

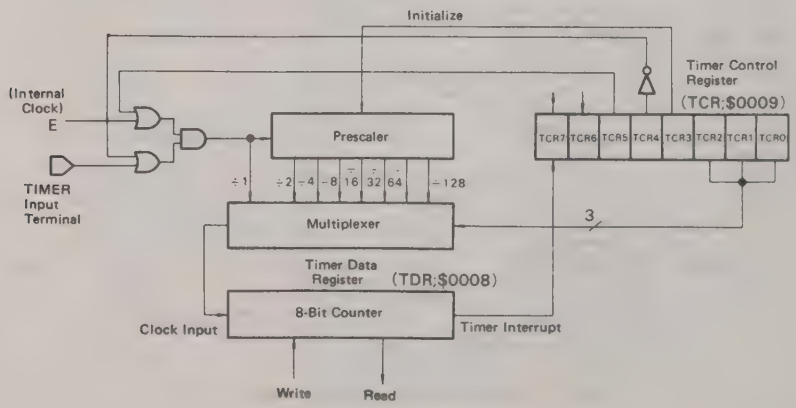


Figure 14 Timer Block Diagram

A prescaler division ratio is selected by the combination of three bits (bits 0, 1 and 2) of the timer control register (see Table 3). There are eight different division ratios: ÷1, ÷2, ÷4, ÷8, ÷16, ÷32, ÷64 and ÷128. After reset, the TCR is set to the ÷1 mode.

Table 3 Prescaler Division Ratio Selection

TCR			Prescaler division ratio
Bit 2	Bit 1	Bit 0	
0	0	0	÷1
0	0	1	÷2
0	1	0	÷4
0	1	1	÷8
1	0	0	÷16
1	0	1	÷32
1	1	0	÷64
1	1	1	÷128

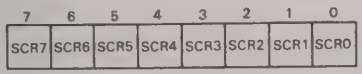
A timer interrupt is enabled when the timer interrupt mask bit is "0", and disabled when the bit is "1". When a timer interrupt occurs, "1" is set in the timer interrupt request bit. This bit can be cleared by writing "0" in that bit.

■SERIAL COMMUNICATION INTERFACE (SCI)

This interface is used for serial transmission or reception of 8-bit data. Sixteen transfer rates are available in the range from 1 μs to approx. 32 ms (for oscillation at 4 MHz).

The SCI consists of three registers, one eighth counter and one prescaler. (See Fig. 15.) SCI communicates with the CPU via the data bus, and with the outside world through bits 5, 6 and 7 of port C. Described below are the operations of each register and data transfer.

●SCI Control Register (SCR; $0010)



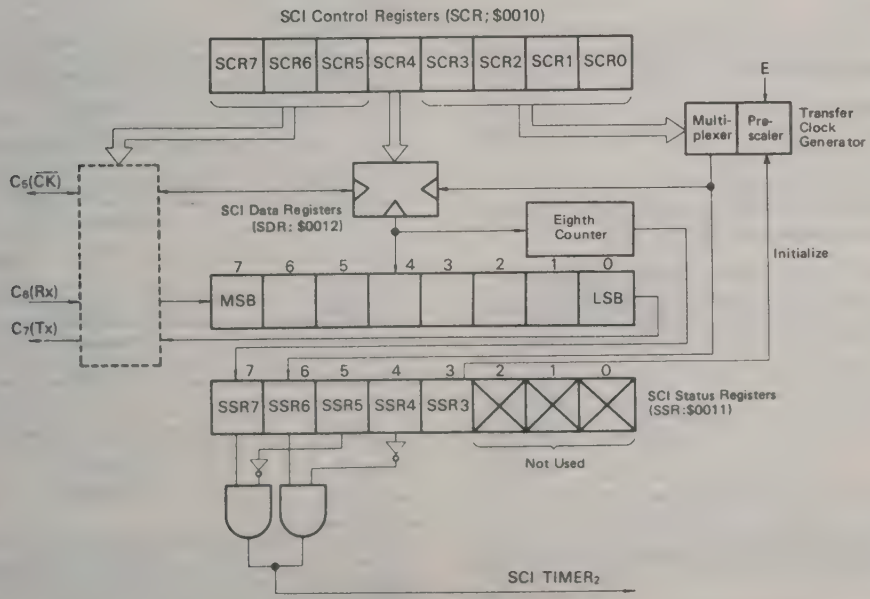


Figure 15 SCI Block Diagram

SCR7	C_7 terminal
0	Used as I/O terminal (by DDR).
1	Serial data output (DDR output)

SCR6	C_6 terminal
0	Used as I/O terminal (by DDR).
1	Serial data input (DDR input)

SCR5	SCR4	Clock source	C_5 terminal
0	0	–	Used as I/O terminal (by DDR).
0	1	–	
1	0	Internal	Clock output (DDR output)
1	1	External	Clock input (DDR input)

Bit 7 (SCR7)

When this bit is set, the DDR corresponding to the C_7 becomes "1" and this terminal serves for output of SCI data. After reset, the bit is cleared to "0".

Bit 6 (SCR6)

When this bit is set, the DDR corresponding to the C_6 becomes "0" and this terminal serves for input of SCI data. After reset, the bit is cleared to "0".

Bits 5 and 4 (SCR5, SCR4)

These bits are used to select a clock source. After reset, the bits are cleared to "0".

Bits 3 ~ 0 (SCR3 ~ SCR0)

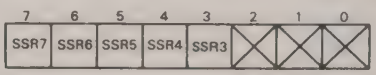
These bits are used to select a transfer clock rate. After reset, the bits are cleared to "0".

SCR3	SCR2	SCR1	SCR0	Transfer clock rate	
				4.00 MHz	4.194 MHz
0	0	0	0	1 μs	0.95 μs
0	0	0	1	2 μs	1.91 μs
0	0	1	0	4 μs	3.82 μs
0	0	1	1	8 μs	7.64 μs
≀	≀	≀	≀	≀	≀
1	1	1	1	32768 μs	1/32 s

•SCI Data Register (SDR; $0012)

A serial-parallel conversion register that is used fot transfer of data.

•SCI Status Register (SSR; $0011)



Bit 7 (SSR7)

Bit 7 is the SCI interrupt request bit which is set upon completion of transmitting or receiving 8-bit data. It is cleared when reset or data is written to or read from the SCI data register with the SCR5="1". The bit can also be cleared by writing "0" in it.

Bit 6 (SSR6)

Bit 6 is the $TIMER_2$ interrupt request bit. $TIMER_2$ is used commonly with the serial clock generator, and SSR6 is set each time the internal transfer clock falls. When reset, the bit is cleared. It also be cleared by writing "0" in it. (For details, see $TIMER_2$.)

Bit 5 (SSR5)

Bit 5 is the SCI interrupt mask bit which can be set or cleared by software. When it is "1", the SCI interrupt (SSR7) is masked. When reset, it is set to "1".

Bit 4 (SSR4)

Bit 4 is the $TIMER_2$ interrupt mask bit which can be set or cleared by software. When the bit is "1", the $TIMER_2$ interrupt (SSR6) is masked. When reset, it is set to "1".

Bit 3 (SSR3)

When "1" is written in this bit, the prescaler of the transfer clock generator is initialized. When read, the bit always is "0".

Bits 2 ~ 0

Not used.

SSR7	SCI interrupt request
0	Absent
1	Present

SSR6	$TIMER_2$ interrupt request
0	Absent
1	Present

SSR5	SCI interrupt mask
0	Enabled
1	Disabled

SSR4	$TIMER_2$ interrupt mask
0	Enabled
1	Disabled

• Data Transmission

By writing the desired control bits into the SCI control registers, a transfer rate and a source of transfer clock are determined and bits 7 and 5 of port C are set at the serial data output terminal and the serial clock terminal, respectively. The transmit data should be stored from the accumulator or index register into the SCI data register. The data written in the SCI data register is output from the C_7/Tx terminal, starting with the LSB, synchronously with the falling edge of the serial clock. (See Fig. 16.) When 8 bit of

data have been transmitted, the interrupt request bit is set in bit 7 of the SCI status register with the rising edge of the last serial clock. This request can be masked by setting bit 5 of the SCI status register. Once the data has been sent, the 8th bit data (MSB) stays at the C_7/Tx terminal. If an external clock source has been selected, the transfer rate determined by bits 0 ~ 3 of the SCI control register is ignored, and the $C_5/\overline{CK}$ terminal is set as input. If the internal clock has been selected, the $C_5/\overline{CK}$ terminal is set as output and clocks are output at the transfer rate selected by bits 0 ~ 3 of the SCI control register.

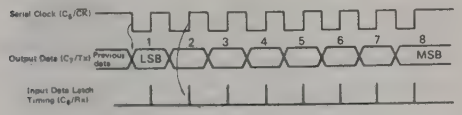


Figure 16 SCI Timing Chart

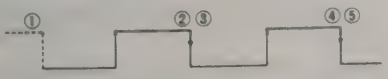
• Data Reception

By writing the desired control bits into the SCI control register, a transfer rate and a source of transfer clock are determined and bit 6 and 5 of port C are set at the serial data input terminal and the serial clock terminal, respectively. Then dummy-writing or -reading the SCI data register, the system is ready for receiving data. (This procedure is not needed after reading the subsequent received data. It must be taken after reset and after not reading the subsequent received.)

The data from the C_6/Rx terminal is input to the SCI data register synchronously with the rising edge of the serial clock (see Fig. 16). When 8 bits of data have been received, the interrupt request bit is set in bit 7 of the SCI status register. This request can be masked by setting bit 5 of the SCI status register. If an external clock source have been selected, the transfer rate determined by bits 0 ~ 3 of the SCI control register is ignored and the data is received synchronously with the clock from the $C_5/\overline{CK}$ terminal. If the internal clock has been selected, the $C_5/\overline{CK}$ terminal is set as output and clocks are output at the transfer rate selected by bits 0 ~ 3 of the SCI control register.

• TIMER$_2$

The SCI transfer clock generator can be used as a timer. The clock selected by bits 3 ~ 0 of the SCI control register (4μs~ approx. 32 ms (for oscillation at 4 MHz)) is input to bit 6 of the SCI status register and the TIMER$_2$ interrupt request bit is set at each falling edge of the clock. Since interrupt requests occur periodically, TIMER$_2$ can be used as a reload counter or clock.



① : Transfer clock generator is reset and mask bit (bit 4 of SCI status register) is cleared.
②, ④ : TIMER$_2$ interrupt request
③, ⑤ : TIMER$_2$ interrupt request bit cleared

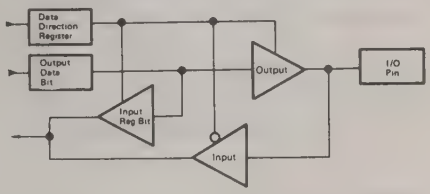
TIMER$_2$ is commonly used with the SCI transfer clock generator. If wanting to use TIMER$_2$ independently of the SCI, specify "External" (SCR5 = 1, SCR4 = 1) as the SCI clock source.

If "Internal" is selected as the clock source, reading or writing the SDR causes the prescaler of the transfer clock generator to be initialized.

■ I/O PORTS

There are 24 input/output terminals (ports A, B, C). Each I/O terminal can be selected for either input or output by the data direction register. More specifically, an I/O port will be input if "0" is written in the data direction register, and output if "1" is written in the data direction register. Port A, B or C reads latched data if it has been programmed as output, even with the output level being fluctuated by the output load. (See Fig. 17.)

When reset, the data direction register and data register go to "0" and all the input/output terminals are used as input.



Bit of data direction register	Bit of output data	Status of output	Input to CPU
1	0	0	0
1	1	1	1
0	X	3-state	Pin

Figure 17 Input/Output Port Diagram

Seven input-only terminals are available (port D). Writing to an input terminal is invalid.

All input/output terminals and input terminals are TTL compatible and CMOS compatible in respect of both input and output.

If I/O ports or input ports are not used, they should be connected to V_{SS} via resistors. With none connected to these terminals, there is the possibility of power being consumed despite that they are not used.

■ RESET

The MPU can be reset either by external reset input ($\overline{RES}$) or power-on reset. (See Fig. 18.) On power up, the reset input must be held "Low" for at least tosc to assure that the internal oscillator is stabilized. A sufficient time of delay can be obtained by connecting a capacitance to the $\overline{RES}$ input as shown in Fig. 19.

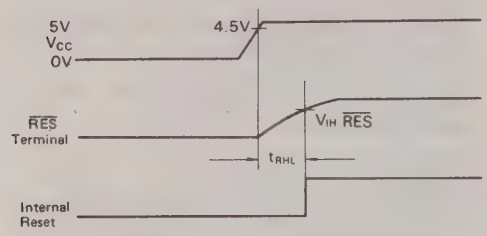


Figure 18 Power On and Reset Timing

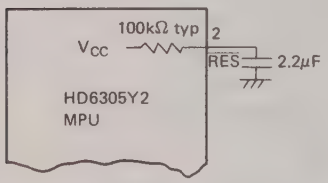


Figure 19 Input Reset Delay Circuit

■INTERNAL OSCILLATOR

The internal oscillator circuit is designed to meet the requirement for minimum external configurations. It can be driven by connecting a crystal (AT cut 2.0 ~ 8.0MHz) or ceramic oscillator between pins 5 and 6 depending on the required oscillation frequency stability.

Three different terminal connections are shown in Fig. 20. Figs. 21 and 22 illustrate the specifications and typical arrangement of the crystal, respectively.

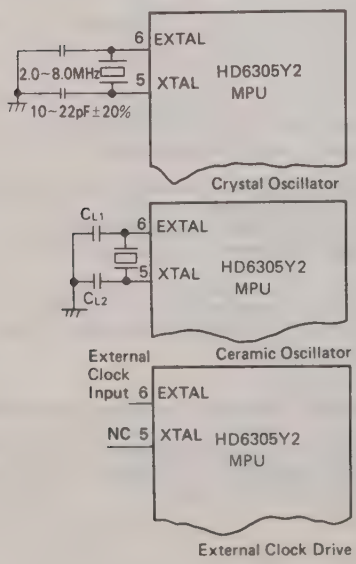


Figure 20 Internal Oscillator Circuit

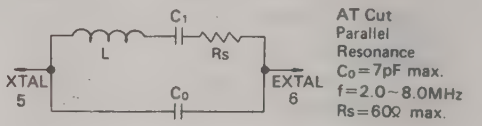
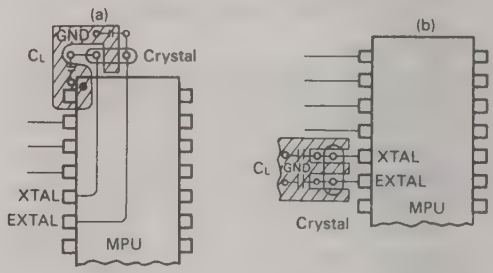


Figure 21 Parameters of Crystal



[NOTE] Use as short wirings as possible for connection of the crystal with the EXTAL and XTAL terminals. Do not allow these wirings to cross others.

Figure 22 Typical Crystal Arrangement

■LOW POWER DISSIPATION MODE

The HD6305Y2 has three low power dissipation modes: wait, stop and standby.

• Wait Mode

When WAIT instruction being executed, the MPU enters into the wait mode. In this mode, the oscillator stays active but the internal clock stops. The CPU stops but the peripheral functions – the timer and the serial communication interface – stay active. (NOTE: Once the system has entered the wait mode, the serial communication interface can no longer be retriggered.) In the wait mode, the registers, RAM and I/O terminals hold their condition just before entering into the wait mode.

The escape from this mode can be done by interrupt ($\overline{INT}$, TIMER/$\overline{INT_2}$ or SCI/TIMER$_2$), $\overline{RES}$ or $\overline{STBY}$. The $\overline{RES}$ resets the MPU and the $\overline{STBY}$ brings it into the standby mode. (This will be mentioned later.)

When interrupt is requested to the CPU and accepted, the wait mode escapes, then the CPU is brought to the operation mode and vectors to the interrupt routine. If the interrupt is masked by the I bit of the condition code register, after releasing from the wait mode the MPU executes the instruction next to the WAIT. If an interrupt other than the $\overline{INT}$ (i.e., TIMER/$\overline{INT_2}$ or SCI/TIMER$_2$) is masked by the timer control

register, miscellaneous register or serial status register, there is no interrupt request to the CPU, so the wait mode cannot be released.

Fig. 23 shows a flowchart for the wait function.

• Stop Mode

When STOP instruction being executed, MPU enters into the stop mode. In this mode, the oscillator stops and the CPU and peripheral functions become inactive but the RAM, registers and I/O terminals hold their condition just before entering into the stop mode.

The escape from this mode can be done by an external interrupt ($\overline{INT}$ or $\overline{INT_2}$), $\overline{RES}$ or $\overline{STBY}$. The $\overline{RES}$ resets the MPU and the $\overline{STBY}$ brings into the standby mode.

When interrupt is requested to the CPU and accepted, the stop mode escapes, then the CPU is brought to the operation mode and vectors to the interrupt routine. If the interrupt is masked by the I bit of the condition code register, after releasing from the stop mode, the MPU executes the instruction next to the STOP. If the $\overline{INT_2}$ interrupt is masked by the miscellaneous register, there is no interrupt request to the MPU, so the stop mode cannot be released.

Fig. 24 shows a flowchart for the stop function. Fig. 25 shows a timing chart of return to the operation mode from the stop mode.

For releasing from the stop mode by an interrupt, oscillation starts upon input of the interrupt and, after the internal delay time for stabilized oscillation, the CPU becomes active.

For restarting by $\overline{RES}$, oscillation starts when the $\overline{RES}$ goes "0" and the CPU restarts when the $\overline{RES}$ goes "1". The duration of $\overline{RES}$="0" must exceed 30 ms to assure stabilized oscillation.

• Standby Mode

The MPU enters into the standby mode when the $\overline{STBY}$ terminal goes "Low". In this mode, all operations stop and the internal condition is reset but the contents of the RAM are hold. The I/O terminals turn to high-impedance state. The standby mode should escape by bringing $\overline{STBY}$ "High". The CPU must be restarted by reset. The timing of input signals at the $\overline{RES}$ and $\overline{STBY}$ terminals is shown in Fig. 26.

Table 4 lists the status of each parts of the MPU in each low power dissipation modes. Transitions between each mode are shown in Fig. 27.

(Note) When I bit of condition code register is "1" and interrupt ($\overline{INT}$, TIMER/$\overline{INT_2}$, SCI/TIMER$_2$) is held, MPU does not enter WAIT mode by the execution of WAIT instruction.
In that case, after the 4 dummy cycles MPU executes the next instruction.
In the same way, when external interrupts ($\overline{INT}$, $\overline{INT_2}$) are held at the bit I set, MPU does not enter STOP mode by the execution of STOP instruction. In that case, also, MPU executes the next instruction after the 4 dummy cycles.

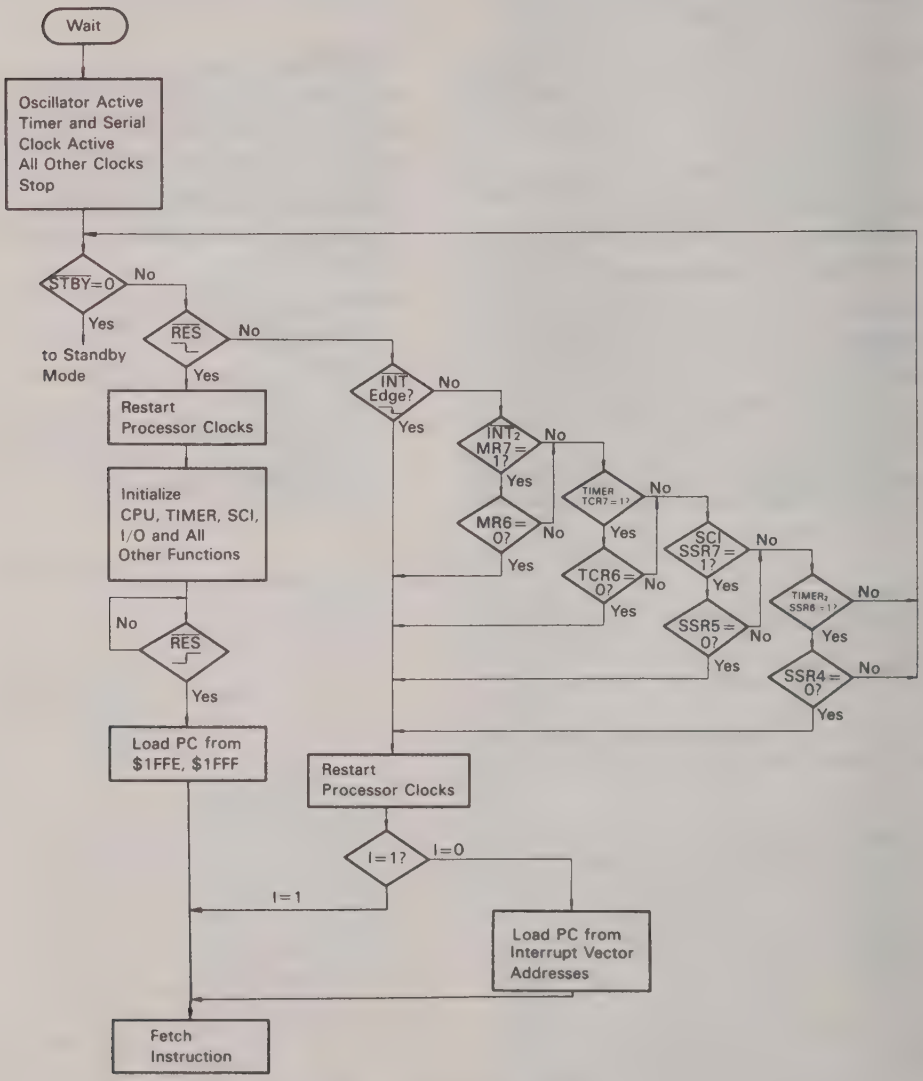


Figure 23 Wait Mode Flow Chart

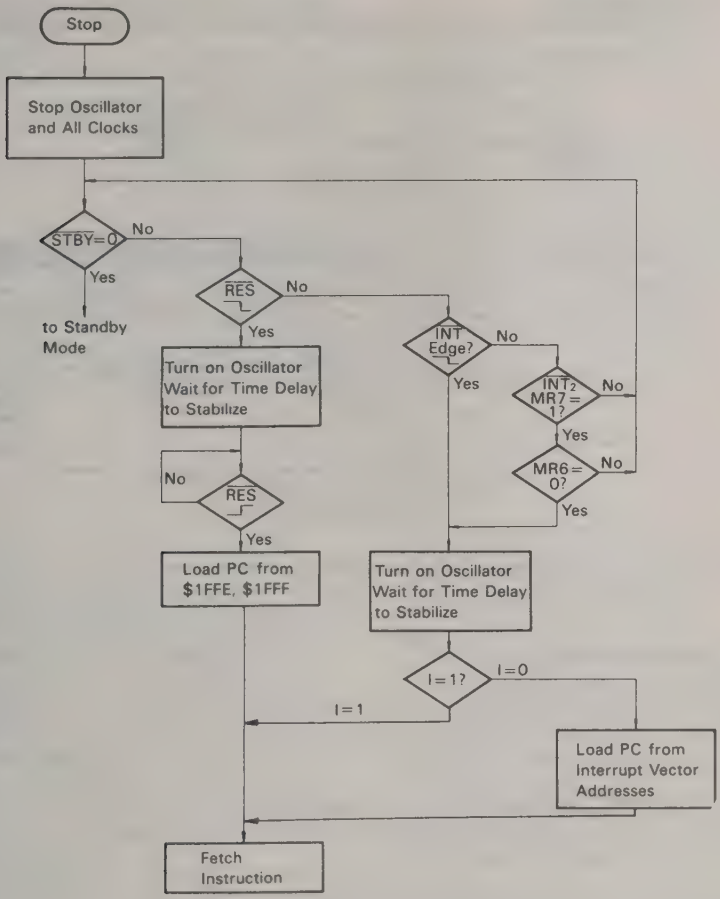
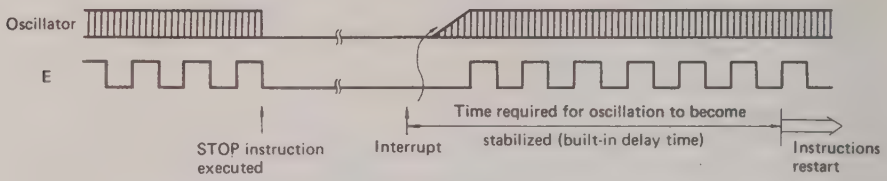
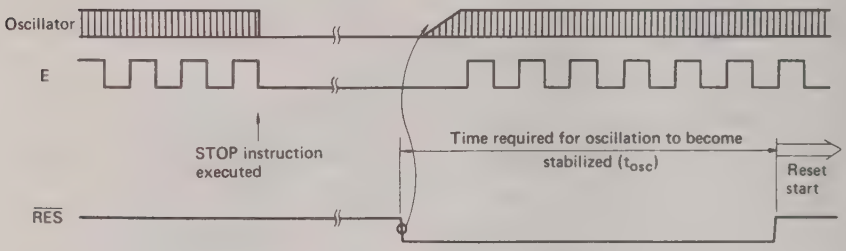


Figure 24 Stop Mode Flow Chart



(a) Restart by Interrupt



(b) Restart by Reset

Figure 25 Timing Chart of Releasing from Stop Mode

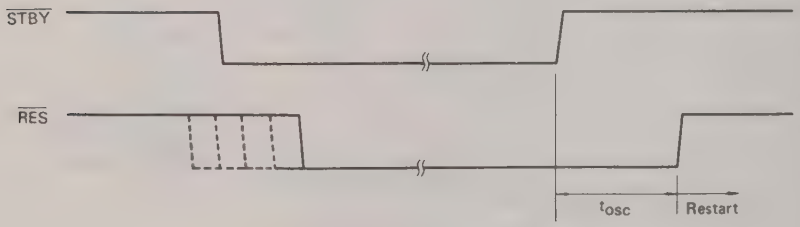


Figure 26 Timing Chart of Releasing from Standby Mode

Table 4 Status of Each Part of MPU in Low Power Dissipation Modes

Mode	Start		Condition						Escape
			Oscillator	CPU	Timer, Serial	Register	RAM	I/O terminal	
WAIT	Software	WAIT instruction	Active	Stop	Active	Keep	Keep	Keep	$\overline{STBY}$, $\overline{RES}$, $\overline{INT}$, $\overline{INT_2}$, each interrupt request of TIMER, $TIMER_2$, SCI
STOP		STOP instruction	Stop	Stop	Stop	Keep	Keep	Keep	$\overline{STBY}$, $\overline{RES}$, $\overline{INT}$, $\overline{INT_2}$
Standby	Hardware	$\overline{STBY}$="Low"	Stop	Stop	Stop	Reset	Keep	High impedance	$\overline{STBY}$="High"

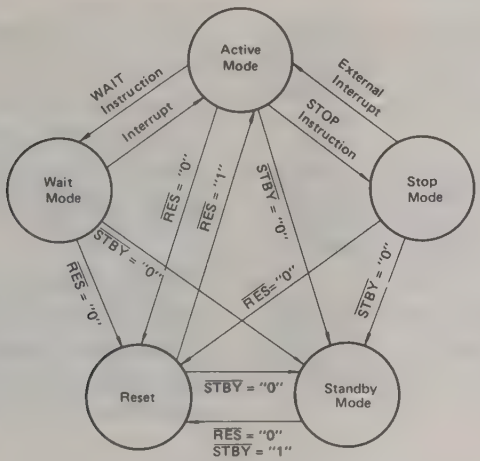


Figure 27 Transitions among Active Mode, Wait Mode, Stop Mode, Standby Mode and Reset

■BIT MANIPULATION

The MPU can use a single instruction (BSET or BCLR) to set or clear one bit of the RAM within page 0 or an I/O port (except the write-only registers such as the data direction register). Every bit of memory or I/O within page 0 ($00 ~ $FF) can be tested by the BRSET or BRCLR instruction; depending on the result of the test, the program can branch to required destinations. Since bits in the RAM on page 0, or I/O can be manipulated, the user may use a bit within the RAM on page 0 as a flag or handle a single I/O bit as an independent I/O terminal. Fig. 28 shows an example of bit manipulation and the validity of test instructions. In the example, the program is configured assuming that bit 0 of port A is connected to a zero cross detector circuit and bit 1 of the same port to the trigger of a triac.

The program shown can activate the triac within a time of 10μs from zero-crossing through the use of only 7 bytes on the memory. The on-chip timer provides a required time of delay and pulse width modulation of power is also possible.

```
          :
SELF 1.   BRCLR 0, PORT A, SELF 1
          BSET 1, PORT A
          BCLR 1, PORT A
          :
```

Figure 28 Example of Bit Manipulation

■ADDRESSING MODES

Ten different addressing modes are available to the MPU.

• Immediate

See Fig. 29. The immediate addressing mode provides access to a constant which does not vary during execution of the program.

This access requires an instruction length of 2 bytes. The effective address (EA) is PC and the operand is fetched from the byte that follows the operation code.

• Direct

See Fig. 30. In the direct addressing mode, the address of the operand is contained in the 2nd byte of the instruction. The user can gain direct access to memory up to the lower 255th address. 192 byte RAM and I/O registers are on page 0 of address space so that the direct addressing mode may be utilized.

• Extended

See Fig. 31. The extended addressing is used for referencing to all addresses of memory. The EA is the contents of the 2 bytes that follow the operation code. An extended addressing instruction requires a length of 3 bytes.

• Relative

See Fig. 32. The relative addressing mode is used with branch instructions only. When a branch occurs, the program counter is loaded with the contents of the byte following the operation code. EA = (PC) + 2 + Rel., where Rel. indicates a signed 8-bit data following the operation code. If no branch occurs, Rel. = 0. When a branch occurs, the program jumps to any byte in the range +129 to −127. A branch instruction requires a length of 2 bytes.

• Indexed (No Offset)

See Fig. 33. The indexed addressing mode allows access up to the lower 255th address of memory. In this mode, an instruction requires a length of one byte. The EA is the contents of the index register.

• Indexed (8-bit Offset)

See Fig. 34. The EA is the contents of the byte following the operation code, plus the contents of the index register. This mode allows access up to the lower 511th address of memory. Each instruction when used in the index addressing mode (8-bit offset) requires a length of 2 bytes.

• Indexed (16-bit Offset)

See Fig. 35. The contents of the 2 bytes following the operation code are added to content of the index register to compute the value of EA. In this mode, the complete memory can be accessed. When used in the indexed addressing mode (16-bit offset), an instruction must be 3 bytes long.

• Bit Set/Clear

See Fig. 36. This addressing mode is applied to the BSET and BCLR instructions that can set or clear any bit on page 0. The lower 3 bits of the operation code specify the bit to be set or cleared. The byte that follows the operation code indicates an address within page 0.

• Bit Test and Branch

See Fig. 37. This addressing mode is applied to the BRSET and BRCLR instructions that can test any bit within page 0 and can be branched in the relative addressing mode. The byte to be tested is addressed depending on the contents of the byte following the operation code. Individual bits within the byte to be tested are specified by the lower 3 bits of the operation code. The 3rd byte represents a relative value which will be added to the program counter when a branch condition is established. Each of these instructions should be 3 bytes long. The value of the test bit is written in the carry bit of the condition code register.

• Implied

See Fig. 38. This mode involves no EA. All information needed for execution of an instruction is contained in the operation code. Direct manipulation on the accumulator and index register is included in the implied addressing mode. Other instructions such as SWI and RTI are also used in this mode. All instructions used in the implied addressing mode should have a length of one byte.

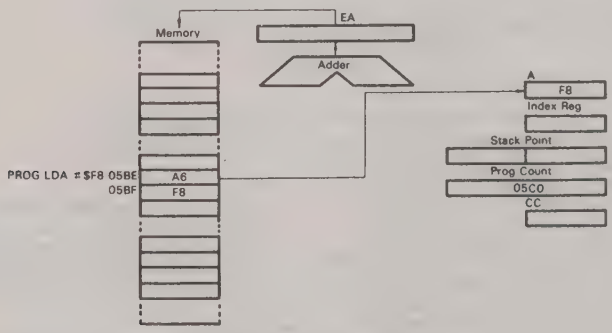


Figure 29 Example of Immediate Addressing

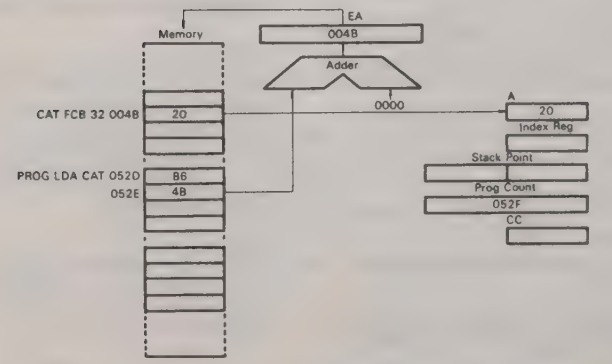


Figure 30 Example of Direct Addressing

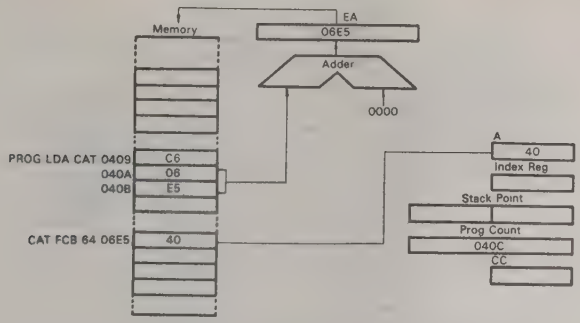


Figure 31 Example of Extended Addressing

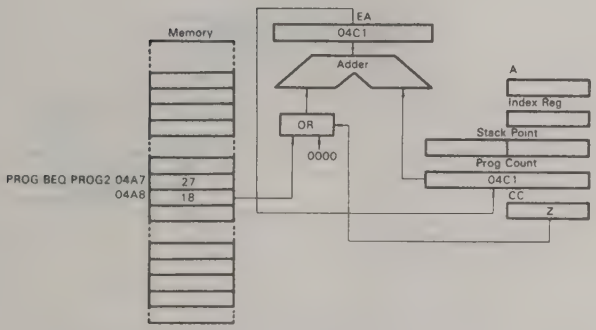


Figure 32 Example of Relative Addressing

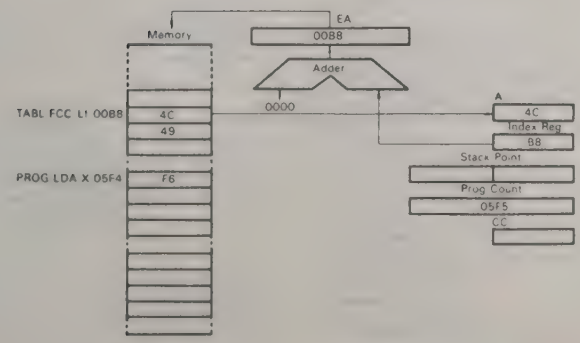


Figure 33 Example of Indexed (No Offset) Addressing

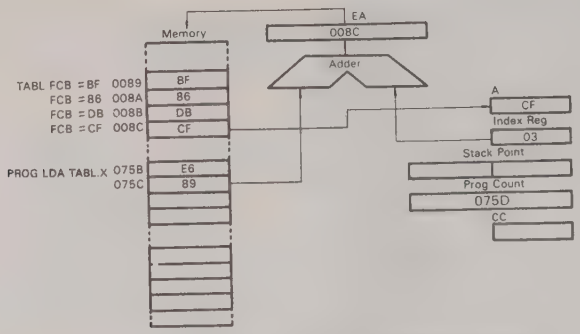


Figure 34 Example of Index (8-bit Offset) Addressing

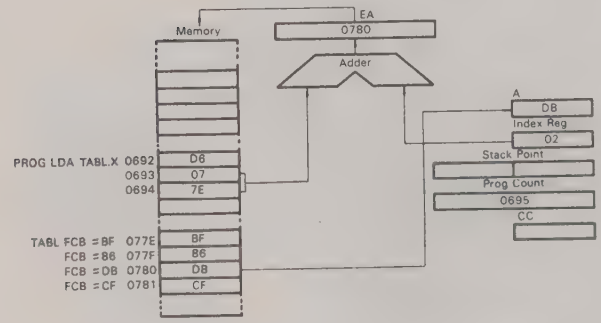


Figure 35 Example of Index (16-bit Offset) Addressing

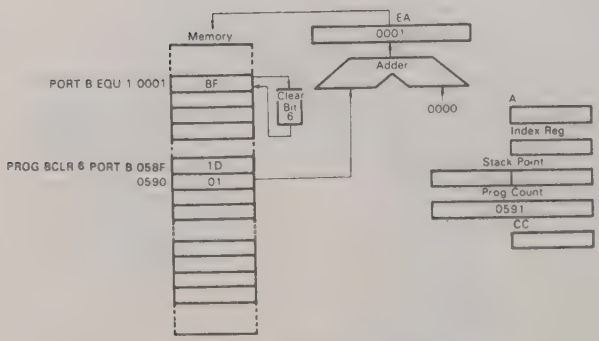


Figure 36 Example of Bit Set/Clear Addressing

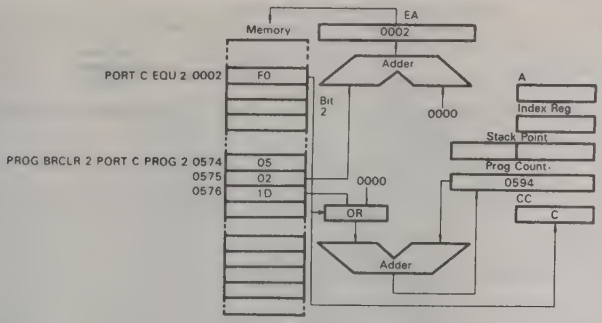


Figure 37 Example of Bit Test and Branch Addressing

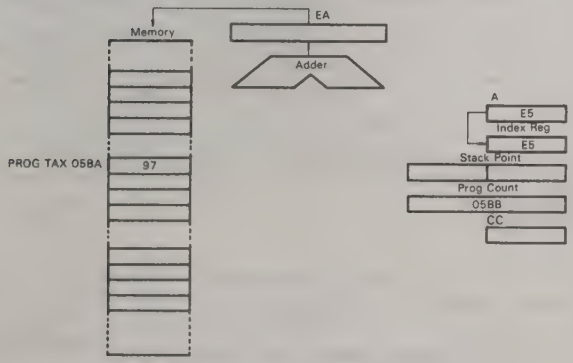


Figure 38 Example of Implied Addressing

■INSTRUCTION SET

There are 62 basic instructions available to the HD6305Y2 MPU. They can be classified into five categories: register/memory, read/modify/write, branch, bit manipulation, and control. The details of each instruction are described in Tables 5 through 11.

• Register/Memory Instructions

Most of these instructions use two operands. One operand is either an accumulator or index register. The other is derived from memory using one of the addressing modes used on the HD6305Y2 MPU. There is no register operand in the unconditional jump instruction (JMP) and the subroutine jump instruction (JSR). See Table 5.

• Read/Modify/Write Instructions

These instructions read a memory or register, then modify or test its contents, and write the modified value into the memory or register. Zero test instruction (TST) does not write data, and is handled as an exception in the read/modify/write group. See Table 6.

• Branch Instructions

A branch instruction branches from the program sequence in progress if a particular condition is established. See Table 7.

• Bit Manipulation Instructions

These instructions can be used with any bit located up to the lower 255th address of memory. Two groups are available; one for setting or clearing and the other for bit testing and branching. See Table 8.

• Control Instructions

The control instructions control the operation of the MPU which is executing a program. See Table 9.

• List of Instructions in Alphabetical Order

Table 10 lists all the instructions used on the HD6305Y2 MPU in the alphabetical order.

• Operation Code Map

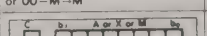
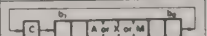
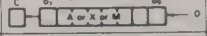
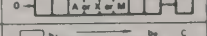
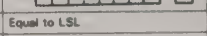
Table 11 shows the operation code map for the instructions used on the MPU.

Table 5 Register/Memory Instructions

Operations	Mnemonic	Addressing Modes: Immediate OP	#	~	Direct OP	#	~	Extended OP	#	~	Indexed (No Offset) OP	#	~	Indexed (8-Bit Offset) OP	#	~	Indexed (16-Bit Offset) OP	#	~	Boolean/ Arithmetic Operation	Condition Code H	I	N	Z	C
Load A from Memory	LDA	A6	2	2	B6	2	3	C6	3	4	F6	1	3	E6	2	4	D6	3	5	M→A	•	•	^	^	•
Load X from Memory	LDX	AE	2	2	BE	2	3	CE	3	4	FE	1	3	EE	2	4	DE	3	5	M →X	•	•	^	^	•
Store A in Memory	STA	–	–	–	B7	2	3	C7	3	4	F7	1	4	E7	2	4	D7	3	5	A →M	•	•	^	^	•
Store X in Memory	STX	–	–	–	BF	2	3	CF	3	4	FF	1	4	EF	2	4	DF	3	5	X→M	•	•	^	^	•
Add Memory to A	ADD	AB	2	2	BB	2	3	CB	3	4	FB	1	3	EB	2	4	DB	3	5	A+M→A	^	•	^	^	^
Add Memory and Carry to A	ADC	A9	2	2	B9	2	3	C9	3	4	F9	1	3	E9	2	4	D9	3	5	A+M+C →A	^	•	^	^	^
Subtract Memory	SUB	A0	2	2	B0	2	3	C0	3	4	F0	1	3	E0	2	4	D0	3	5	A−M→A	•	•	^	^	^
Subtract Memory from A with Borrow	SBC	A2	2	2	B2	2	3	C2	3	4	F2	1	3	E2	2	4	D2	3	5	A−M−C →A	•	•	^	^	^
AND Memory to A	AND	A4	2	2	B4	2	3	C4	3	4	F4	1	3	E4	2	4	D4	3	5	A · M→A	•	•	^	•	•
OR Memory with A	ORA	AA	2	2	BA	2	3	CA	3	4	FA	1	3	EA	2	4	DA	3	5	A+M →A	•	•	^	^	•
Exclusive OR Memory with A	EOR	A8	2	2	B8	2	3	C8	3	4	F8	1	3	E8	2	4	D8	3	5	A⊕M→A	•	•	^	^	•
Arithmetic Compare A with Memory	CMP	A1	2	2	B1	2	3	C1	3	4	F1	1	3	E1	2	4	D1	3	5	A−M	•	•	^	•	•
Arithmetic Compare X with Memory	CPX	A3	2	2	B3	2	3	C3	3	4	F3	1	3	E3	2	4	D3	3	5	X−M	•	•	^	•	•
Bit Test Memory with A (Logical Compare)	BIT	A5	2	2	B5	2	3	C5	3	4	F5	1	3	E5	2	4	D5	3	5	A · M	•	•	^	•	•
Jump Unconditional	JMP	–	–	–	BC	2	2	CC	3	3	FC	1	2	EC	2	3	DC	3	4		•	•	•	•	•
Jump to Subroutine	JSR	–	–	–	BD	2	5	CD	3	6	FD	1	5	ED	2	5	DD	3	6		•	•	•	•	•

Symbols: Op = Operation
= Number of bytes
~ = Number of cycles

Table 6 Read/Modify/Write Instructions

Operations	Mnemonic	Addressing Modes: Implied(A) OP	#	~	Implied(X) OP	#	~	Direct OP	#	~	Indexed (No Offset) OP	#	~	Indexed (8-Bit Offset) OP	#	~	Boolean/Arithmetic Operation	Condition Code H	I	N	Z	C
Increment	INC	4C	1	2	5C	1	2	3C	2	5	7C	1	5	6C	2	6	A+1 →A or X+1→X or M+1 →M	•	•	^	^	•
Decrement	DEC	4A	1	2	5A	1	2	3A	2	5	7A	1	5	6A	2	6	A−1 →A or X−1→X or M−1→M	•	•	^	^	•
Clear	CLR	4F	1	2	5F	1	2	3F	2	5	7F	1	5	6F	2	6	00 →A or 00 →X or 00 →M	•	•	0	1	•
Complement	COM	43	1	2	53	1	2	33	2	5	73	1	5	63	2	6	$\overline{A}$ →A or $\overline{X}$ →X or $\overline{M}$ →M	•	•	^	^	1
Negate (2's Complement)	NEG	40	1	2	50	1	2	30	2	5	70	1	5	60	2	6	00−A→A or 00−X→X or 00−M→M	•	•	^	^	^
Rotate Left Thru Carry	ROL	49	1	2	59	1	2	39	2	5	79	1	5	69	2	6		•	•	^	^	^
Rotate Right Thru Carry	ROR	46	1	2	56	1	2	36	2	5	76	1	5	66	2	6		•	•	^	^	^
Logical Shift Left	LSL	48	1	2	58	1	2	38	2	5	78	1	5	68	2	6		•	•	^	^	^
Logical Shift Right	LSR	44	1	2	54	1	2	34	2	5	74	1	5	64	2	6		•	•	0	^	^
Arithmetic Shift Right	ASR	47	1	2	57	1	2	37	2	5	77	1	5	67	2	6		•	•	^	^	^
Arithmetic Shift Left	ASL	48	1	2	58	1	2	38	2	5	78	1	5	68	2	6	Equal to LSL	•	•	^	^	^
Test for Negative or Zero	TST	4D	1	2	5D	1	2	3D	2	4	7D	1	4	6D	2	5	A−00 or X−00 or M−00	•	•	^	^	•

Symbols: Op = Operation
= Number of bytes
~ = Number of cycles

Table 7 Branch Instructions

Operations	Mnemonic	Addressing Modes			Branch Test	Condition Code				
		Relative								
		OP	#	~		H	I	N	Z	C
Branch Always	BRA	20	2	3	None	•	•	•	•	•
Branch Never	BRN	21	2	3	None	•	•	•	•	•
Branch IF Higher	BHI	22	2	3	C+Z=0	•	•	•	•	•
Branch IF Lower or Same	BLS	23	2	3	C+Z=1	•	•	•	•	•
Branch IF Carry Clear	BCC	24	2	3	C=0	•	•	•	•	•
(Branch IF Higher or Same)	(BHS)	24	2	3	C=0	•	•	•	•	•
Branch IF Carry Set	BCS	25	2	3	C=1	•	•	•	•	•
(Branch IF Lower)	(BLO)	25	2	3	C=1	•	•	•	•	•
Branch IF Not Equal	BNE	26	2	3	Z=0	•	•	•	•	•
Branch IF Equal	BEQ	27	2	3	Z=1	•	•	•	•	•
Branch IF Half Carry Clear	BHCC	28	2	3	H=0	•	•	•	•	•
Branch IF Half Carry Set	BHCS	29	2	3	H=1	•	•	•	•	•
Branch IF Plus	BPL	2A	2	3	N=0	•	•	•	•	•
Branch IF Minus	BMI	2B	2	3	N=1	•	•	•	•	•
Branch IF Interrupt Mask Bit is Clear	BMC	2C	2	3	I=0	•	•	•	•	•
Branch IF Interrupt Mask Bit is Set	BMS	2D	2	3	I=1	•	•	•	•	•
Branch IF Interrupt Line is Low	BIL	2E	2	3	INT=0	•	•	•	•	•
Branch IF Interrupt Line is High	BIH	2F	2	3	INT=1	•	•	•	•	•
Branch to Subroutine	BSR	AD	2	5	——	•	•	•	•	•

Symbols: Op = Operation
= Number of bytes
~ = Number of cycles

Table 8 Bit Manipulation Instructions

Operations	Mnemonic	Addressing Modes						Boolean/ Arithmetic Operation	Branch Test	Condition Code				
		Bit Set Clear			Bit Test and Branch									
		OP	#	~	OP	#	~			H	I	N	Z	C
Branch IF Bit n is set	BRSET n(n=0···7)				2·n	3	5	——	Mn=1	•	•	•	•	^
Branch IF Bit n is clear	BRCLR n(n=0···7)				01+2·n	3	5	——	Mn=0	•	•	•	•	^
Set Bit n	BSET n(n=0···7)	10+2·n	2	5	–		–	1→Mn	——	•	•	•	•	•
Clear Bit n	BCLR n(n=0···7)	11+2·n	2	5	–		–	0→Mn	——	•	•	•	•	•

Symbols: Op = Operation
= Number of bytes
~ = Number of cycles

Table 9 Control Instructions

Operations	Mnemonic	Addressing Modes: Implied OP	#	~	Boolean Operation	Condition Code: H	I	N	Z	C
Transfer A to X	TAX	97	1	2	A→X	●	●	●	●	●
Transfer X to A	TXA	9F	1	2	X→A	●	●	●	●	●
Set Carry Bit	SEC	99	1	1	1→C	●	●	●	●	1
Clear Carry Bit	CLC	98	1	1	0→C	●	●	●	●	0
Set Interrupt Mask Bit	SEI	9B	1	2	1→I	●	1	●	●	●
Clear Interrupt Mask Bit	CLI	9A	1	2	0→I	●	0	●	●	●
Software Interrupt	SWI	83	1	10		●	1	●	●	●
Return from Subroutine	RTS	81	1	5		●	●	●	●	●
Return from Interrupt	RTI	80	1	8		?	?	?	?	?
Reset Stack Pointer	RSP	9C	1	2	$FF→SP	●	●	●	●	●
No-Operation	NOP	9D	1	1	Advance Prog. Cntr Only	●	●	●	●	●
Decimal Adjust A	DAA	8D	1	2	Converts binary add of BCD charcters into BCD format	●	●	^	^	^ *
Stop	STOP	8E	1	4		●	●	●	●	●
Wait	WAIT	8F	1	4		●	●	●	●	●

Symbols: Op = Operation
= Number of bytes
~ = Number of cycles

* Are BCD characters of upper byte 10 or more? (They are not cleared if set in advance.)

Table 10 Instruction Set (in Alphabetical Order)

Mnemonic	Addressing Modes: Implied	Immediate	Direct	Extended	Relative	Indexed (No Offset)	Indexed (8-Bit)	Indexed (16-Bit)	Bit Set/ Clear	Bit Test & Branch	Condition Code: H	I	N	Z	C
ADC		×	×	×		×	×	×			^	●	^	^	^
ADD		×	×	×		×	×	×			^	●	^	^	^
AND		×	×	×		×	×	×			●	●	^	^	●
ASL	×		×			×	×				●	●	^	^	^
ASR	×		×			×	×				●	●	^	^	^
BCC					×						●	●	●	●	●
BCLR									×		●	●	●	●	●
BCS					×						●	●	●	●	●
BEQ					×						●	●	●	●	●
BHCC					×						●	●	●	●	●
BHCS					×						●	●	●	●	●
BHI					×						●	●	●	●	●
(BHS)					×						●	●	●	●	●
BIH					×						●	●	●	●	●
BIL					×						●	●	●	●	●
BIT		×	×	×		×	×	×			●	●	^	^	●
(BLO)					×						●	●	●	●	●
BLS					×						●	●	●	●	●
BMC					×						●	●	●	●	●
BMI					×						●	●	●	●	●
BMS					×						●	●	●	●	●
BNE					×						●	●	●	●	●
BPL					×						●	●	●	●	●
BRA					×						●	●	●	●	●

(to be continued)

Condition Code Symbols:

H	Half Carry (From Bit 3)	C	Carry/Borrow
I	Interrupt Mask	^	Test and Set if True, Cleared Otherwise
N	Negative (Sign Bit)	●	Not Affected
Z	Zero	?	Load CC Register From Stack

Table 10 Instruction Set (in Alphabetical Order)

Mnemonic	Addressing Modes										Condition Code				
	Implied	Immediate	Direct	Extended	Relative	Indexed (No Offset)	Indexed (8-Bit)	Indexed (16-Bit)	Bit Set Clear	Bit Test & Branch	H	I	N	Z	C
BRN					×						•	•	•	•	•
BRCLR										×	•	•	•	•	^
BRSET										×	•	•	•	•	^
BSET									×		•	•	•	•	•
BSR					×						•	•	•	•	•
CLC	×										•	•	•	•	0
CLI	×										•	0	•	•	•
CLR	×		×			×	×				•	•	0	1	•
CMP		×	×	×		×	×	×			•	•	^	^	^
COM	×		×			×	×				•	•	^	^	1
CPX		×	×	×		×	×	×			•	•	^	^	^
DAA	×										•	•	^	^	^
DEC	×		×			×	×				•	•	^	^	•
EOR		×	×	×		×	×	×			•	•	^	^	•
INC	×		×			×	×				•	•	^	^	•
JMP			×	×		×	×	×			•	•	•	•	•
JSR			×	×		×	×	×			•	•	•	•	•
LDA		×	×	×		×	×	×			•	•	^	^	•
LDX		×	×	×		×	×	×			•	•	^	^	•
LSL	×		×			×	×				•	•	^	^	^
LSR	×		×			×	×				•	•	0	^	^
NEG	×		×			×	×				•	•	^	^	^
NOP	×										•	•	•	•	•
ORA		×	×	×		×	×	×			•	•	^	^	•
ROL	×		×			×	×				•	•	^	^	^
ROR	×		×			×	×				•	•	^	^	^
RSP	×										•	•	•	•	•
RTI	×										?	?	?	?	?
RTS	×										•	•	•	•	•
SBC		×	×	×		×	×	×			•	•	^	^	^
SEC	×										•	•	•	•	1
SEI	×										•	1	•	•	•
STA			×	×		×	×	×			•	•	^	^	•
STOP	×										•	•	•	•	•
STX			×	×		×	×	×			•	•	^	^	•
SUB		×	×	×		×	×	×			•	•	^	^	^
SWI	×										•	1	•	•	•
TAX	×										•	•	•	•	•
TST	×		×			×	×				•	•	^	^	•
TXA	×										•	•	•	•	•
WAIT	×										•	•	•	•	•

Condition Code Symbols:

H	Half Carry (From Bit 3)	C	Carry Borrow
I	Interrupt Mask	^	Test and Set if True, Cleared Otherwise
N	Negative (Sign Bit)	•	Not Affected
Z	Zero	?	Load CC Register From Stack

Table 11 Operation Code Map

	Bit Manipulation		Branch	Read/Modify/Write					Control		Register/Memory						
	Test & Branch	Set/ Clear	Rel	DIR	A	X	,X1	,X0	IMP	IMP	IMM	DIR	EXT	,X2	,X1	,X0	
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	← HIGH
0	BRSET0	BSET0	BRA	NEG					RTI*	–	SUB						0
1	BRCLR0	BCLR0	BRN	–					RTS*	–	CMP						1
2	BRSET1	BSET1	BHI	–					–	–	SBC						2
3	BRCLR1	BCLR1	BLS	COM					SWI*	–	CPX						3
4	BRSET2	BSET2	BCC	LSR					–	–	AND						4
5	BRCLR2	BCLR2	BCS	–							BIT						5
6	BRSET3	BSET3	BNE	ROR						–	LDA						6
7	BRCLR3	BCLR3	BEQ	ASR						TAX*			STA			STA(+1)	7
8	BRSET4	BSET4	BHCC	LSL/ASL						CLC	EOR						8
9	BRCLR4	BCLR4	BHCS	ROL					–	SEC	ADC						9
A	BRSET5	BSET5	BPL	DEC						CLI*	ORA						A
B	BRCLR5	BCLR5	BMI							SEI*	ADD						B
C	BRSET6	BSET6	BMC	INC					–	RSP*	–	JMP(−1)					C
D	BRCLR6	BCLR6	BMS	TST(−1)	TST			TST(−1)	DAA*	NOP	BSR*	JSR(+2)		JSR(+1)		JSR(+2)	D
E	BRSET7	BSET7	BIL	–					STOP*	–	LDX						E
F	BRCLR7	BCLR7	BIH	CLR					WAIT*	TXA*		STX				STX(+1)	F
	3/5	2/5	2/3	2/5	1/2	1/2	2/6	1/5	1/*	1/1	2/2	2/3	3/4	3/5	2/4	1/3	

LOW

(NOTES) 1. "–" is an undefined operation code.

2. The lowermost numbers in each column represent a byte count and the number of cycles required (byte count/number of cycles). The number of cycles for the mnemonics asterisked (*) is as follows:

RTI	8	TAX	2
RTS	5	RSP	2
SWI	10	TXA	2
DAA	2	BSR	5
STOP	4	CLI	2
WAIT	4	SEI	2

3. The parenthesized numbers must be added to the cycle count of the particular instruction.

- **Additional Instructions**

The following new instructions are used on the HD6305Y2:

DAA Converts the contents of the accumulator into BCD code.

WAIT Causes the MPU to enter the wait mode. For this mode, see the topic, Wait Mode.

STOP Causes the MPU to enter the stop mode. For this mode, see the topic, Stop Mode.

■ OPERATION AT EACH INSTRUCTION CYCLE

The HD6305Y2 employs a mechanism of the pipeline control for the instruction fetch and the subsequent instruction fetch is performed during the current instruction being executed.

Table 12 provides the information about the relationship among each data on the Address Bus, Data Bus and R/$\overline{W}$ status in cycle-by-cycle basis during the execution of each instruction.

Table 12 Cycle-by-Cycle Operation

Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/$\overline{W}$	Data Bus
IMMEDIATE					
ADC, ADD, AND, BIT, CMP, CPX, EOR, LDA, LDX, ORA, SBC, SUB	2	1	Op Code Address +1	1	Operand Data
		2	Op Code Address +2	1	Next Op Code
DIRECT					
ADC, ADD, AND, BIT, CMP, CPX, EOR, LDA, LDX, ORA, SBC, SUB	3	1	Op Code Address +1	1	Address of Operand
		2	Address of Operand	1	Operand Data
		3	Op Code Address +2	1	Next Op Code

(to be continued)

Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/$\overline{W}$	Data Bus
STA, STX	3	1	Op Code Address +1	1	Address of Operand
		2	Address of Operand	0	Data from Acc. / Data from Ix.
		3	Op Code Address +1	1	Next Op Code
JMP	2	1	Op Code Address +1	1	Jump Address
		2	Jump Address	1	Next Op Code
JSR	5	1	Op Code Address +1	1	Jump Address (LSB)
		2	1FFF	1	Irrelevant Data
		3	Stack Pointer	0	Return Address (LSB)
		4	Stack Pointer -1	0	Return Address (MSB)
		5	Jump Address	1	First Subroutine Op Code
ASR, CLR, COM,	5	1	Op Code Address +1	1	Address of Operand
DEC, INC, LSL,		2	Address of Operand	1	Operand Data
LSR, NEG, ROL,		3	1FFF	1	Irrelevant Data
ROR		4	Address of Operand	0	New Operand Data
		5	Op Code Address +2	1	Next Op Code
TST	4	1	Op Code Address +1	1	Address of Operand
		2	Address of Operand	1	Operand Data
		3	1FFF	1	Irrelevant Data
		4	Op Code Address +2	1	Next Op Code
EXTENDED					
ADC, ADD, AND,	4	1	Op Code Address +1	1	Address of Operand (MSB)
BIT, CMP, CPX,		2	Op Code Address +2	1	Address of Operand (LSB)
EOR, LDA, LDX,		3	Address of Operand	1	Operand Data
ORA, SBC, SUB		4	Op Code Address +3	1	Next Op Code
STA, STX	4	1	Op Code Address +1	1	Address of Operand (MSB)
		2	Op Code Address +2	1	Address of Operand (LSB)
		3	Address of Operand	0	Data from Acc. / Data from Ix.
		4	Op Code Address +3	1	Next Op Code
JMP	3	1	Op Code Address +1	1	Jump Address (MSB)
		2	Op Code Address +2	1	Jump Address (LSB)
		3	Jump Address	1	Next Op Code
JSR	6	1	Op Code Address +1	1	Jump Address (MSB)
		2	Op Code Address +2	1	Jump Address (LSB)
		3	1FFF	1	Irrelevant Data
		4	Stack Pointer	0	Return Address (LSB)
		5	Stack Pointer -1	0	Return Address (MSB)
		6	Jump Address	1	First Subroutine Op Code
INDEXED (No offset)					
ADC, ADD, AND,	3	1	Op Code Address +1	1	Next Op Code
BIT, CMP, CPX,		2	Ix	1	Operand Data
EOR, LDA, LDX,		3	Op Code Address +1	1	Next Op Code
ORA, SBC, SUB					
STA, STX	4	1	Op Code Address +1	1	Next Op Code
		2	1FFF	1	Irrelevant Data
		3	Ix	0	Data from Acc. / Data from Ix.
		4	Op Code Address +1	1	Next Op Code
JMP	2	1	Op Code Address +1	1	Next Op Code
		2	Ix	1	First Op Code of Jump Routine

(to be continued)

Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/W	Data Bus
JSR	5	1	Op Code Address +1	1	Next Op Code
		2	1FFF	1	Irrelevant Data
		3	Stack Pointer	0	Return Address (LSB)
		4	Stack Pointer -1	0	Return Address (MSB)
		5	Ix	1	First Subroutine Op Code
ASR, CLR, COM,	5	1	Op Code Address +1	1	Next Op Code
DEC, INC, LSL,		2	Ix	1	Operand Data
LSR, NEG, ROL,		3	1FFF	1	Irrelevant Data
ROR		4	Ix	0	New Operand Data
		5	Op Code Address +1	1	Next Op Code
TST	4	1	Op Code Address +1	1	Next Op Code
		2	Ix	1	Operand Data
		3	1FFF	1	Irrelevant Data
		4	Op Code Address +1	1	Next Op Code
INDEXED (8-bit offset)					
ADC, ADD, AND,	4	1	Op Code Address +1	1	Offset
BIT, CMP, CPX,		2	1FFF	1	Irrelevant Data
EOR, LDA, LDX,		3	Ix + Offset	1	Operand Data
ORA, SBC, SUB		4	Op Code Address +2	1	Next Op Code
STA, STX	4	1	Op Code Address +1	1	Offset
		2	1FFF	1	Irrelevant Data
		3	Ix + Offset	0	{ Data from Acc. / Data from Ix.
		4	Op Code Address +2	1	Next Op Code
JMP	3	1	Op Code Address +1	1	Offset
		2	1FFF	1	Irrelevant Data
		3	Ix + Offset	1	First Op Code of Jump Routine
JSR	5	1	Op Code Address +1	1	Offset
		2	1FFF	1	Irrelevant Data
		3	Stack Pointer	0	Return Address (LSB)
		4	Stack Pointer -1	0	Return Address (MSB)
		5	Ix + Offset	1	First Subroutine Op Code
ASR, CLR, COM,	6	1	Op Code Address +1	1	Offset
DEC, INC, LSL,		2	1FFF	1	Irrelevant Data
LSR, NEG, ROL,		3	Ix + Offset	1	Operand Data
ROR		4	1FFF	1	Irrelevant Data
		5	Ix + Offset	0	New Operand Data
		6	Op Code Address +1	1	Next Op Code
TST	5	1	Op Code Address +1	1	Offset
		2	1FFF	1	Irrelevant Data
		3	Ix + Offset	1	Operand Data
		4	1FFF	1	Irrelevant Data
		5	Op Code Address +2	1	Next Op Code
INDEXED (16-bit offset)					
ADC, ADD, AND,	5	1	Op Code Address +1	1	Offset (MSB)
BIT, CMP, CPX,		2	Op Code Address +2	1	Offset (LSB)
EOR, LDA, LDX,		3	1FFF	1	Irrelevant Data
ORA, SBC, SUB		4	Ix + Offset	1	Operand Data
		5	Op Code Address +1	1	Next Op Code

(to be continued)

Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/$\overline{W}$	Data Bus
STA, STX	5	1	Op Code Address +1	1	Offset (MSB)
		2	Op Code Address +2	1	Offset (LSB)
		3	1FFF	1	Irrelevant Data
		4	Ix + Offset	0	Data from Acc. / Data from Ix.
		5	Op Code Address +3	1	Next Op Code
JMP	4	1	Op Code Address +1	1	Offset (MSB)
		2	Op Code Address +2	1	Offset (LSB)
		3	1FFF	1	Irrelevant Data
		4	Ix + Offset	1	First Op Code of Jump Routine
JSR	6	1	Op Code Address +1	1	Offset (MSB)
		2	Op Code Address +2	1	Offset (LSB)
		3	1FFF	1	Irrelevant Data
		4	Stack Pointer	0	Return Address (LSB)
		5	Stack Pointer −1	0	Return Address (MSB)
		6	Ix + Offset	1	First Subroutine Op Code
IMPLIED					
ASR, CLR, COM, DEC, INC, LSL, LSR, NEG, ROL, ROR, TST	2	1	Op Code Address +1	1	Next Op Code
		2	Op Code Address +1	1	Next Op Code
CLC, NOP, SEC	1	1	Op Code Address +1	1	Next Op Code
RSP, TAX, TXA	2	1	Op Code Address +1	1	Next Op Code
		2	Op Code Address +1	1	Next Op Code
CLI, SEI	2	1	Op Code Address +1	1	Next Op Code
		2	1FFF	1	Irrelevant Data
DAA	2	1	Op Code Address +1	1	Next Op Code
		2	Op Code Address +1	1	Next Op Code
STOP, WAIT	4	1	Op Code Address +1	1	Next Op Code
		2	1FFF	1	Irrelevant Data
		3	1FFF	1	Irrelevant Data
		4	Op Code Address +1	1	Next Op Code
RTI	8	1	Op Code Address +1	1	Next Op Code
		2	1FFF	1	Irrelevant Data
		3	Stack Pointer	1	CC
		4	Stack Pointer +1	1	Acc.
		5	Stack Pointer +2	1	Ix.
		6	Stack Pointer +3	1	Return Address (MSB)
		7	Stack Pointer +4	1	Return Address (LSB)
		8	Return Address	1	First Op Code of Return Routine
RTS	5	1	Op Code Address +1	1	Next Op Code
		2	1FFF	1	Irrelevant Data
		3	Stack Pointer	1	Return Address (MSB)
		4	Stack Pointer +1	1	Return Address (LSB)
		5	Return Address	1	First Op Code of Return Routine
SWI	10	1	Op Code Address +1	1	Next Op Code
		2	1FFF	1	Irrelevant Data
		3	Stack Pointer	0	Return Address (LSB)
		4	Stack Pointer−1	0	Return Address (MSB)
		5	Stack Pointer−2	0	Ix.
		6	Stack Pointer−3	0	Acc.
		7	Stack Pointer−4	0	CC
		8	Vector Address 1FFC	1	Address of SWI Routine (MSB)
		9	Vector Address 1FFD	1	Address of SWI Routine (LSB)
		10	Address of SWI Routine	1	First Op Code of SWI Routine

(to be continued)

Address Mode & Instructions	Cycles	Cycle #	Address Bus	R/$\overline{W}$	Data Bus
RELATIVE					
BCC, BCS, BEQ, BHCC, BHCS, BHI, BIH, BIL, BLS, BMC, BMI, BMS, BNE, BPL, BRA, BRN	3	1	Op Code Address +1	1	Next Op Code
		2	1FFF	1	Irrelevant Data
		3	Branch Address .....Test = "1" Op Code Address +1 .... Test = "0"	1	First Op Code of Branch Routine Next Op Code
BSR	5	1	Op Code Address +1	1	Offset
		2	1FFF	1	Irrelevant Data
		3	Stack Pointer	0	Return Address (LSB)
		4	Stack Pointer−1	0	Return Address (MSB)
		5	Branch Address	1	First Op Code of Subroutine
BIT TEST AND BRANCH					
BRCLR, BRSET	5	1	Op Code Address +1	1	Address of Operand
		2	Address of Operand	1	Operand Data
		3	Op Code Address +2	1	Offset
		4	1FFF	1	Irrelevant Data
		5	Branch Address .....Test ="1" Op Code Address +3 .....Test = "0"	1	First Op Code of Branch Address Next Op Code
BIT SET/CLEAR					
BCLR, BSET	5	1	Op Code Address +1	1	Address of Operand
		2	Address of Operand	1	Operand Data
		3	1FFF	1	Irrelevant Data
		4	Address of Operand	0	New Operand Data
		5	Op Code Address +1	1	Next Op Code

HD6800, HD68A00, HD68B00
MPU (Micro Processing Unit)

The HD6800 is a monolithic 8-bit microprocessor forming the central control function for Hitachi's HMCS6800 family. Compatible with TTL, the HD6800 as with all HMCS6800 system parts, requires only one 5V power supply, and no external TTL devices for bus interface. The HD68A00 and HD68B00 are high speed versions.

The HD6800 is capable of addressing 65k bytes of memory with its 16-bit address lines. The 8-bit data bus is bi-directional as well as 3-state, making direct memory addressing and multiprocessing applications realizable.

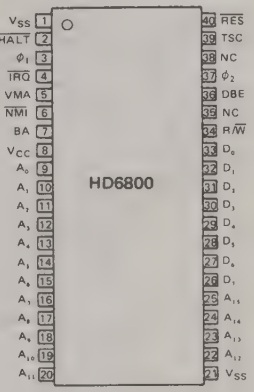
HD6800P, HD68A00P, HD68B00P

(DP-40)

■ FEATURES

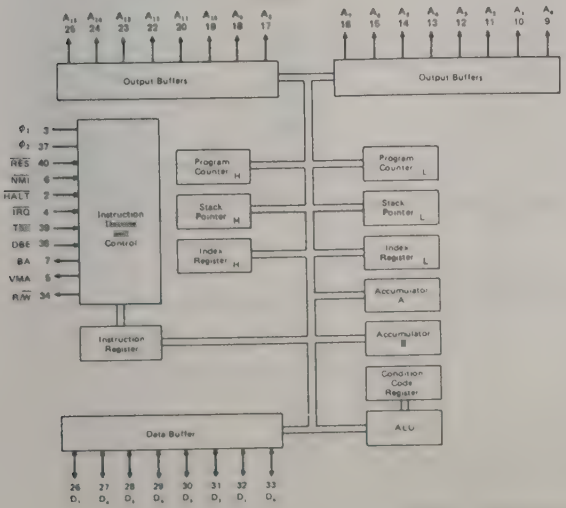
- Versatile 72 Instruction – Variable Length (1~3 Byte)
- Seven Addressing Modes – Direct, Relative, Immediate, Indexed, Extended, Implied and Accumulator
- Variable Length Stack
- Vectored Restart
- Maskable Interrupt
- Separate Non-Maskable Interrupt – Internal Registers Saved in Stack
- Six Internal Registers – Two Accumulators, Index Register, Program Counter, Stack Pointer and Condition Code Register
- Direct Memory Accessing (DMA) and Multiple Processor Capability
- Clock Rates as High as 2.0 MHz (HD6800 ··· 1 MHz, HD68A00 ··· 1.5 MHz, HD68B00 ··· 2.0 MHz)
- Halt and Single Instruction Execution Capability
- Compatible with MC6800, MC68A00 and MC68B00

■ PIN ARRANGEMENT



(Top View)

■ BLOCK DIAGRAM



■ ABSOLUTE MAXIMUM RATINGS

Item	Symbol	Value	Unit
Supply Voltage	V_{CC}*	−0.3 ~ +7.0	V
Input Voltage	V_{in}*	−0.3 ~ +7.0	V
Operating Temperature	T_{opr}	− 20 ~ + 75	°C
Storage Temperature	T_{stg}	− 55 ~ +150	°C

* With respect to V_{SS} (SYSTEM GND)
(NOTE) Permanent LSI damage may occur if maximum rating are exceeded. Normal operation should be under recommended operating conditions. If these conditions are exceeded, it could affect reliability of LSI.

■ RECOMMENDED OPERATING CONDITION

Item	Symbol	min	typ	max	Unit
Supply Voltage	V_{CC}*	4.75	5.0	5.25	V
Input Voltage	V_{IL}*	−0.3	–	0.8	V
	V_{IH}*	2.0	–	V_{CC}	V
Operating Temperature	T_{opr}	−20	25	75	°C

* With respect to V_{SS} (SYSTEM GND)

■ ELECTRICAL CHARACTERISTICS

● DC CHARACTERISTICS (V_{CC} = 5V ± 5%, V_{SS} = 0V, Ta = −20~+75°C, unless otherwise noted.)

Item		Symbol	Test Condition	min	typ*	max	Unit
Input "High" Voltage	Logic**	V_{IH}		2.0	–	V_{CC}	V
Input "Low" Voltage	Logic**	V_{IL}		− 0.3	—	0.8	V
Clock Input "High" Voltage	ϕ_1, ϕ_2	V_{IHC}		V_{CC} − 0.6	–	V_{CC} + 0.3	V
Clock Input "Low" Voltage	ϕ_1, ϕ_2	V_{ILC}		−0.3	–	0.4	V
Output "High" Voltage	D_0~D_7	V_{OH}	I_{OH} = −205μA	2.4	–	–	V
	A_0~A_{15}, R/$\overline{W}$ VMA		I_{OH} = −145μA	2.4	–	–	V
	BA		I_{OH} = −100μA	2.4	–	–	V
Output "Low" Voltage		V_{OL}	I_{OL} = 1.6mA	–	–	0.4	V
Input Leakage Current	Logic***	I_{in}	V_{in} = 0~5.25V, All other pins are connected to GND	-2.5	–	2.5	μA
	ϕ_1, ϕ_2			-100	–	100	μA
Three-State (Off-state) Input Current	D_0~D_7	I_{TSI}	V_{in} = 0.4 ~ 2.4V	-10	–	10	μA
	A_0~A_{15}, R/$\overline{W}$			-100	–	100	μA
Power Dissipation		P_D		–	0.5	1.0	W
Input Capacitance	Logic***	C_{in}	V_{in} = 0V, Ta = 25°C, f = 1 MHz	–	6.5	10	pF
	D_0~D_7			–	10	12.5	pF
	ϕ_1			–	25	35	pF
	ϕ_2			–	45	70	pF
Output Capacitance	A_0~A_{15}, R/$\overline{W}$ VMA, BA	C_{out}	V_{in} = 0V, Ta = 25°C, f = 1 MHz	–	–	12	pF

* Ta = 25°C, V_{CC} = 5V
** All inputs except ϕ_1 and ϕ_2
*** All inputs except ϕ_1, ϕ_2 and D_0~D_7

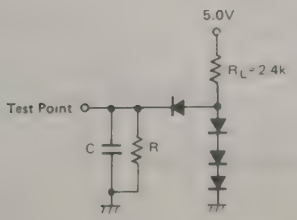
- **AC CHARACTERISTICS** (V_{CC} = 5V ± 5%, V_{SS} = 0V, Ta = -20~+75°C, unless otherwise noted.)

1. TIMING CHARACTERISTICS OF CLOCK PULSE ϕ_1 and ϕ_2

Item		Symbol	Test Condition	HD6800			HD68A00			HD68B00			Unit
				min	typ	max	min	typ	max	min	typ	max	
Frequency of Operation		f		0.1	–	1.0	0.1	–	1.5	0.1	–	2.0	MHz
Cycle Time		t_{cyc}	Fig. 10	1.000	–	10	0.666	–	10	0.500	–	10	μs
Clock Pulse Width	ϕ_1, ϕ_2	PW_{CH1}, PW_{CH2}	Fig. 10	400	–	4,500	230	–	4,500	180	–	4,500	ns
Rise and Fall Times	ϕ_1, ϕ_2	t_r, t_f	Fig. 10	–	–	100	–	–	100	–	–	100	ns
Delay Time (Clock Internal)		t_d	Fig. 10	0	–	4,500	0	–	4,500	0	–	4,500	ns
Clock "High" Level Time		t_{UT}	Fig. 10	900	–	–	600	–	–	440	–	–	ns

2. READ/WRITE CHARACTERISTICS

Item		Symbol	Test Condition	HD6800			HD68A00			HD68B00			Unit
				min	typ	max	min	typ	max	min	typ	max	
Address Delay Time	C=90pF	t_{AD1}	Fig. 11, Fig. 12	–	–	270	–	–	180	–	–	150	ns
	C=30pF	t_{AD2}	Fig. 11, Fig. 12	–	–	250	–	–	165	–	–	135	ns
Data Setup Time (Read)		t_{DSR}	Fig. 11	100	–	–	60	–	–	40	–	–	ns
Peripheral Read Access Time $t_{acc} = t_{UT} - (t_{AD} + t_{DSR})$		t_{acc}	Fig. 11	530	–	–	360	–	–	250	–	–	ns
Input Data Hold Time		t_H	Fig. 11	10	–	–	10	–	–	10	–	–	ns
Output Data Hold Time		t_H	Fig. 12	20	–	–	20	–	–	20	–	–	ns
Address Hold Time (Address, R/$\overline{W}$, VMA)		t_{AH}	Fig. 11, Fig. 12	10	–	–	10	–	–	10	–	–	ns
Enable "High" Time for DBE Input		t_{EH}	Fig. 12	450	–	–	280	–	–	220	–	–	ns
Data Delay Time (Write)		t_{DDW}	Fig. 12	–	–	225	–	–	200	–	–	160	ns
Data Bus Enable Down Time (During ϕ_1 Up Time)		$t_{\overline{DBE}}$	Fig. 12	150	–	–	120	–	–	75	–	–	ns
Data Bus Enable Delay Time		t_{DBED}	Fig. 12	300	–	–	250	–	–	180	–	–	ns
Data Bus Enable Rise and Fall Times		t_{DBEr} t_{DBEf}	Fig. 12	–	–	25	–	–	25	–	–	25	ns
Processor Control Setup Time		t_{PCS}		200	–	–	140	–	–	110	–	–	ns
Processor Control Rise and Fall Times		t_{PCr} t_{PCf}		–	–	100	–	–	100	–	–	100	ns
Bus Available Delay Time (BA)		t_{BA}		–	–	250	–	–	165	–	–	135	ns
Three-State Delay Time		t_{TSD}		–	–	270	–	–	270	–	–	220	ns



C = 130pF for D_0~D_7
= 90pF for A_0~A_{15}, R/$\overline{W}$, and VMA
= 30pF for BA
R = 11kΩ for D_0~D_7
= 16kΩ for A_0~A_{15}, R/$\overline{W}$ and VMA
= 24kΩ for BA
C includes Stray Capacitance.
All diodes are 1S2074 Ⓗ or equivalent

Figure 1 Bus Timing Test Load

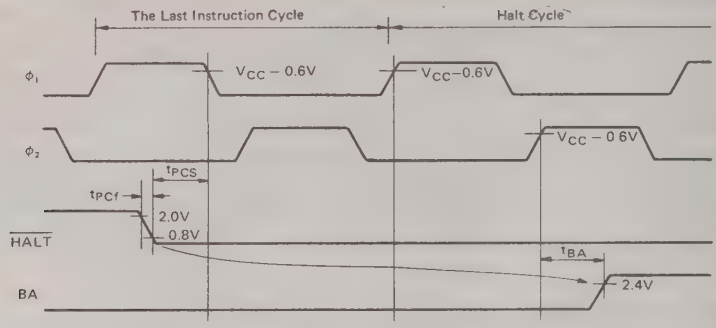


Figure 2 Timing of $\overline{\text{HALT}}$ and BA

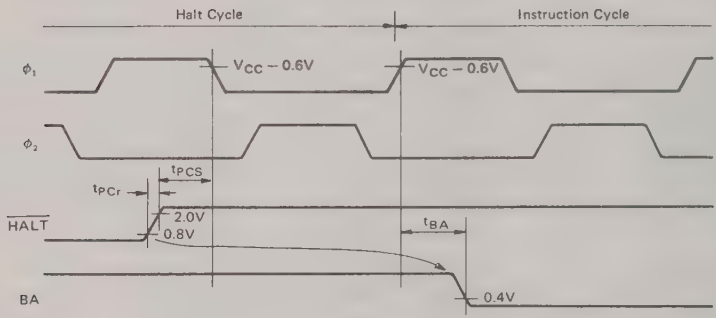


Figure 3 Timing of $\overline{\text{HALT}}$ and BA

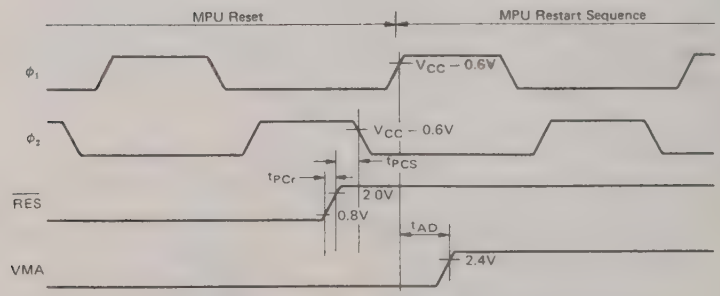


Figure 4 $\overline{\text{RES}}$ and MPU Restart Sequence

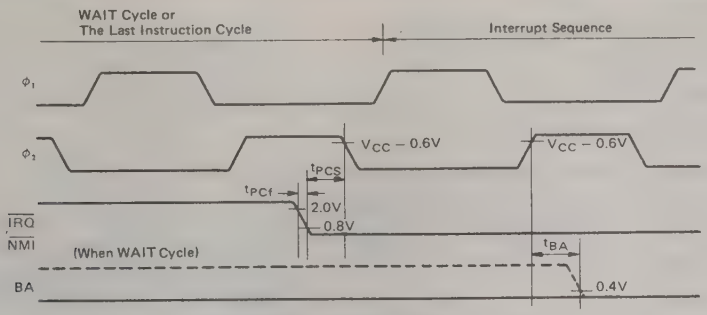


Figure 5 $\overline{IRQ}$ and $\overline{NMI}$ Interrupt Timing

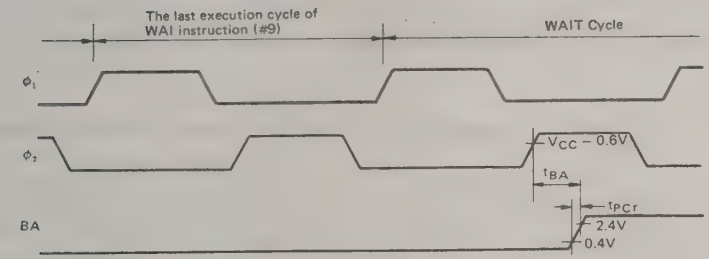


Figure 6 WAI Instruction and BA Timing

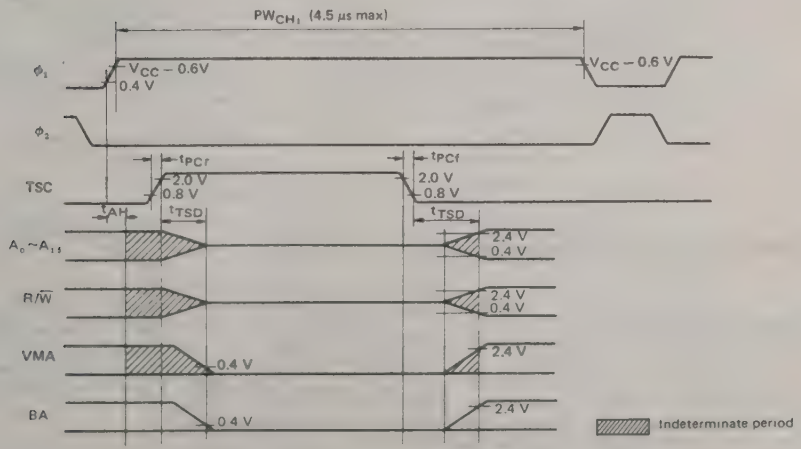


Figure 7 TSC Input and MPU Output

■ MPU REGISTERS

The MPU provides several registers in Fig. 8, which is available for use by the programmer.

Each register is described below.

- **Program Counter (PC)**

The program counter is a two byte (16-bit) register that points to the current program address.

- **Stack Pointer (SP)**

The stack pointer is a two byte register that contains the address of the next available location in an external push-down/pop-up stack. This stack is normally a random access Read/Write memory that may have any location (address) that is convenient. In those applications that require storage of information in the stack when power is lost, the stack must be non-volatile.

- **Index Register (IX)**

The index register is a two byte register that is used to store data or a sixteen bit memory address for the Indexed mode of memory addressing.

- **Accumulators (ACCA, ACCB)**

The MPU contains two 8-bit accumulators that are used to hold operands and results from an arithmetic logic unit (ALU).

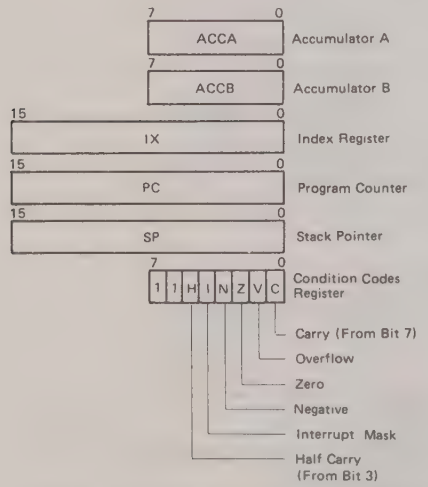


Figure 8 Programming Model of the Microprocessing Unit

- **Condition Code Register (CCR)**

The condition code register indicates the results of an Arithmetic Logic Unit operation: Negative (N), Zero (Z), Overflow (V), Carry from bit 7 (C), and half carry from bit 3(H). These bits of the Condition Code Register are used as testable conditions for the conditional branch instructions. Bit 4 is the interrupt mask bit (I). The unused bits of the Condition Code Register (b6 and b7) are "1". The detail block diagram of the microprossing unit is shown in Fig. 9.

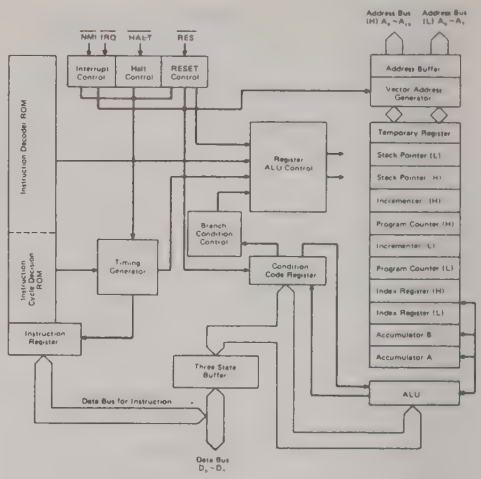


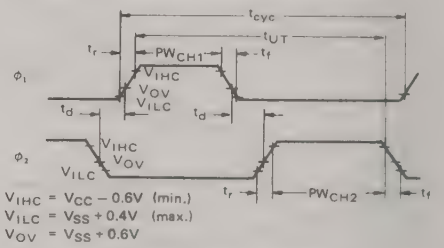
Figure 9 Internal Block Diagram of MPU

■ MPU SIGNAL DESCRIPTION

Proper operations of the MPU requires that certain control and timing signals (Fig. 9) be provided to accomplish specific functions. The functions of pins are explained in this section.

- **Clock (ϕ_2, ϕ_2)**

Two pins are used to provide the clock signals. A two-phase non-overlapping clock is provided as shown in Fig. 10.



$V_{IHC} = V_{CC} - 0.6V$ (min.)
$V_{ILC} = V_{SS} + 0.4V$ (max.)
$V_{OV} = V_{SS} + 0.6V$

Figure 10 Clock Timing Waveform

- **Address Bus ($A_0 \sim A_{15}$)**

Sixteen pins are used for the address bus. The outputs are three-state bus drivers capable of driving one standard TTL load and 90pF. When the output is turned off, it is essentially an open circuit. This permits the MPU to be used in DMA applications. Putting TSC in its high state forces the Address bus to go into the three-state mode.

- **Data Bus ($D_0 \sim D_7$)**

Eight pins are used for the data bus. It is bidirectional, transferring data to and from the memory and peripheral devices. It also has three-state output buffers capable of driving one standard TTL load and 130pF. Data Bus is placed in the three-state mode when DBE is "Low."

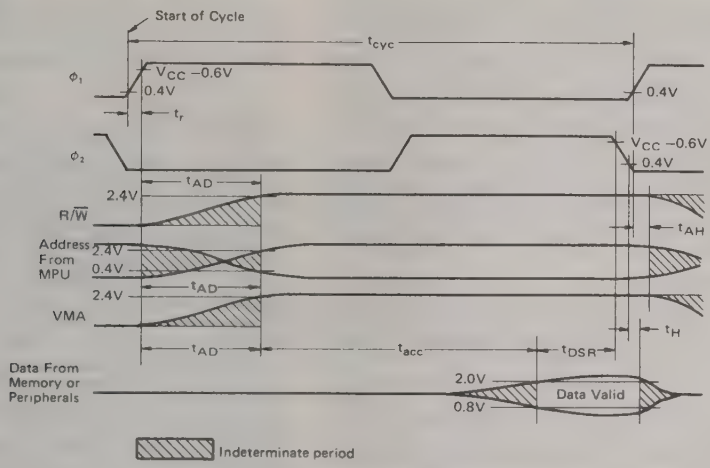


Figure 11 Read from Memory or Peripherals

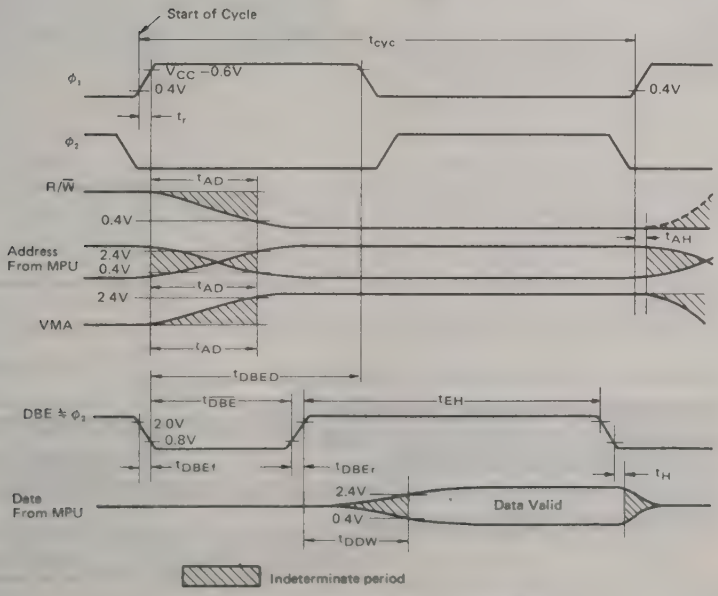


Figure 12 Write to Memory or Peripherals

- **Data Bus Enable (DBE)**

This input is the three-state control signal for the MPU data bus and will enable the bus drivers when in the "High" state; will make the bus driver off when in the "Low" state. This input is TTL compatible; however in normal operation, it would be driven by ϕ_2 clock. During an MPU read cycle, the data bus drivers will be disabled internally. When it is desired that another device control the data bus such as in Direct Memory Access (DMA) applications, DBE should be held "Low."

If additional data setup or hold time is required on an MPU write, the DBE down time can be decreased as shown in Fig. 13 (DBE $\neq \phi_2$). The minimum down time for DBE is $t_{\overline{DBE}}$ as shown and must occur within ϕ_1 up time. As for the characteristical values in Fig. 12, refer to the table of electrical characteristics.

- **Bus Available (BA)**

The BA signal will normally be in the "Low" state. When activated, it will go to the "High" state indicating that the microprocessor has stopped and that the address bus is available. This will occur if the $\overline{HALT}$ line is in the "Low" state or the processor is in the WAIT state as a result of the execution of a WAIT instruction. At such time, all three-state output drivers will go to their off state and other outputs to their normally inactive level. The processor is removed from the WAIT state by the occurrence of a maskable (mask bit I = 0) or nonmaskable interrupt. This output is capable of driving one standard TTL load and 30pF. If TSC is in the "High" state, Bus Available will be "Low".

- **Read/Write (R/$\overline{W}$)**

This TTL compatible output signals the peripherals and memory devices whether the MPU is in a Read ("High") or Write ("Low") state. The normal standby state of this signal is Read ("High"). Three-State Control going "High" will turn R/$\overline{W}$ to the off (high impedance) state. Also, when the processor is halted, it will be in the off state. This output is capable of driving one standard TTL load and 90pF.

- **Reset ($\overline{RES}$)**

The $\overline{RES}$ input is used to reset and start the MPU from a power down condition resulting from a power failure or initial start-up of the processor. This input can also be used to re-initialize the machine at any time after start-up.

If a "High" level is detected in this input, this will signal the MPU to begin the reset sequence. During the reset sequence, the contents of the last two locations (FFFE, FFFF) in memory will be loaded into the Program Counter to point to the beginning of the reset routine. During the reset routine, the interrupt mask bit is set and must be cleared under program control before the MPU can be interrupted by $\overline{IRQ}$. While $\overline{RES}$ is "Low" (assuming a minimum of 8 clock cycles have occured) the MPU output signals will be in the following states; VMA = "Low", BA = "Low", Data Bus = high impedance, R/$\overline{W}$ = "High" (read state), and the Address Bus will contain the reset address FFFE. Fig. 13 illustrates a power up sequence using the Reset control line. After the power supply reaches 4.75V, a minimum of eight clock cycles are required for the processor to stabilize in preparation for restarting. During these eight cycles, VMA will be in an indeterminate state so any devices that are enabled by VMA which could accept a false write during this time (such as a battery-backed RAM) must be disabled until VMA is forced "Low" after eight cycles. $\overline{RES}$ can go "High" asynchronously with the system clock any time after the eighth cycle.

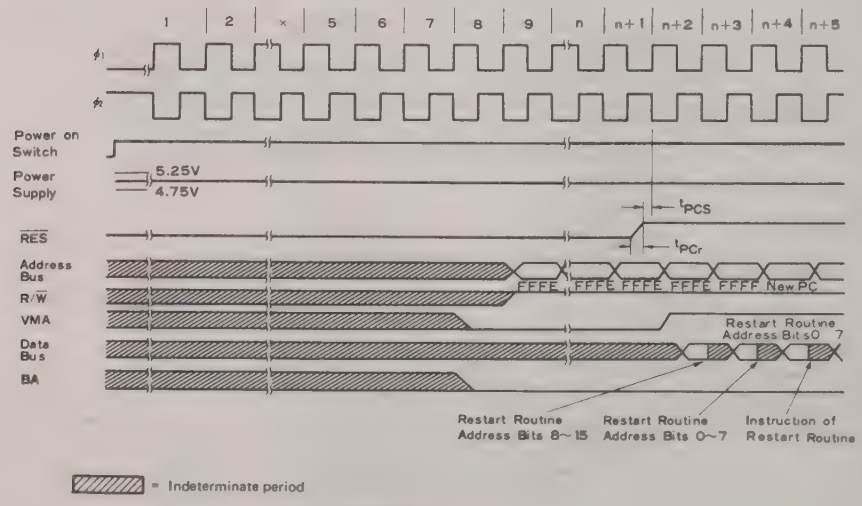


Figure 13 $\overline{RES}$ Timing

The Reset control line may also be used to reinitialize the MPU system at any time during its operation. This is accomplished by pulsing $\overline{RES}$ "Low" for the duration of a minimum of three complete ϕ_2 cycles. The $\overline{RES}$ pulse can be completely asynchronous with the MPU system clock and will be recognized during ϕ_2 if setup time t_{PCS} is met.

- **Interrupt Request ($\overline{IRQ}$)**

This level sensitive input requests that an interrupt sequence be generated within the machine. The processor will wait until it completes the current instruction that is being executed before it recognizes the request. If the interrupt mask bit in the Condition Code Register is not set, the machine will begin an interrupt sequence. The Program Counter, Index Register, Accumulators, and Condition Code Register are stored away on the stack.

Next the MPU will respond to the interrupt request by setting the interrupt mask bit "1" so that no further interrupts may occur. At the end of the cycle, a 16-bit address will be loaded that points to a vectoring address which is located in memory locations FFF8 and FFF9. An address loaded at these locations causes the MPU to branch to an interrupt routine in memory. Interrupt timing is shown in Fig. 14.

The $\overline{HALT}$ line must be in the "High" state for interrupts to be serviced. Interrupts will be latched internally while $\overline{HALT}$ is "Low". The $\overline{IRQ}$ has a high impedance pullup device internal to the chip; however a 3kΩ external resistor to V_{CC} should be used for wire-OR and optimum control of interrupts.

- **Non-Maskable Interrupt ($\overline{NMI}$) and Wait for Interrupt (WAI)**

The MPU is capable of handling two types of interrupts: maskable ($\overline{IRQ}$) as described earlier, and non-maskable ($\overline{NMI}$). $\overline{IRQ}$ is maskable by the interrupt mask in the Condition Code Register while $\overline{NMI}$ is not maskable. The handling of these interrupts by the MPU is the same except that each has its own vector address. The behavior of the MPU when interrupted is shown in Fig. 14 which details the MPU response to an interrupt while the MPU is executing the control program. The interrupt shown could be either $\overline{IRQ}$ or $\overline{NMI}$ and can be asynchronous with respect to ϕ_2. The interrupt is shown going "Low" at time t_{PCS} in cycle #0 which precedes the first cycle of an instruction (OP code fetch). This instruction is not executed but instead the Program Counter (PC), Index Register (IX), Accumulators (ACCX), and the Condition Code Register (CCR) are pushed onto the stack.

The Interrupt Mask bit is set to prevent further interrupts. The address of the interrupt service routine is then fetched from FFFC, FFFD for an $\overline{NMI}$ interrupt and from FFF8, FFF9 for an $\overline{IRQ}$ interrupt. Upon completion of the interrupt service routine, the execution of RTI will pull the PC, IX, ACCX, and CCR off of the stack; the Interrupt Mask bit is restored to its condition prior to interrupts. Fig. 15 is a similar interrupt sequence, except in this case, a WAIT instruction has been executed in preparation for the interrupt. This technique speeds up the MPU's response to the interrupt because the stacking of

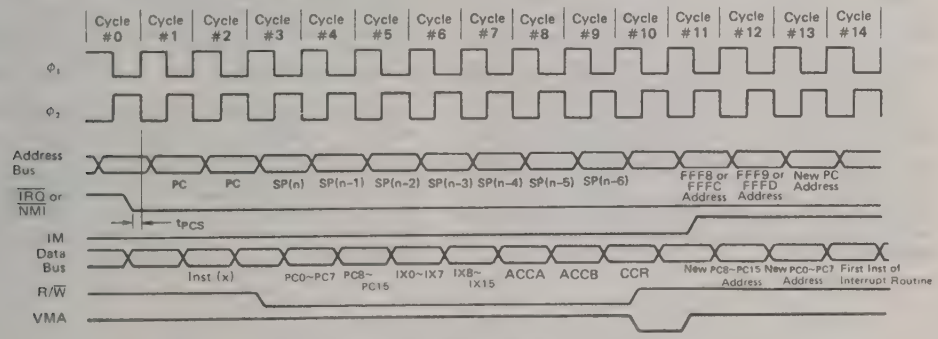
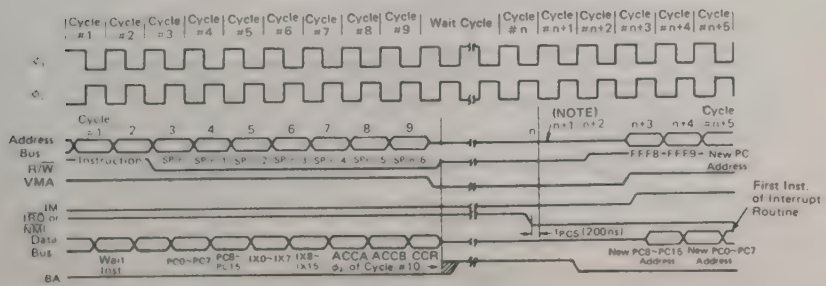


Figure 14 Interrupt Timing



(NOTE) Midrange waveform indicates high impedance state.

Figure 15 WAI Instruction Timing

the PC, IX, ACCX, and the CCR is already done.

While the MPU is waiting for the interrupt, Bus Available will go "High" indicating the following states of the control lines: VMA is "Low", and the Address Bus, R/$\overline{W}$ and Data Bus are all in the high impedance state. After the interrupt occurs, it is serviced as previously described.

Table 1 Memory Map for Interrupt Vectors

Vector		Description
MS	LS	
FFFE	FFFF	Restart
FFFC	FFFD	Non-maskable Interrupt
FFFA	FFFB	Software Interrupt
FFF8	FFF9	Interrupt Request

Refer to Figure 18 for program flow for Interrupts.

- **Three State Control (TSC)**

When the Three State Control (TSC) line is "High" level, the Address Bus and the R/$\overline{W}$ line are placed in a high impedance State. VMA and BA are forced "Low" when TSC = "High" to prevent false reads or writes on any device enabled by VMA. It is necessary to delay program execution while TSC is held "High". This is done by insuring that no transitions of ϕ_1 (or ϕ_2) occur during this period. (Logic levels of the clocks are irrelevant so long as they do not change.)

Since the MPU is a dynamic device, the ϕ_1 clock can be stopped for a maximum time PW_{CH1} without destroying data within the MPU. TSC then can be used in a short Direct Memory Access (DMA) application.

Fig. 16 shows the effect of TSC on the MPU. The Address Bus and R/$\overline{W}$ line will reach the high impedance state at t_{TSD} (three-state delay), with VMA being forced "Low". In this example, the Data Bus is also in the high impedance state while ϕ_2 is being held "Low" since DBE=ϕ_2. At this point in time, a DMA transfer could occur on cycles #3 and #4. When TSC is returned "Low," the MPU address and R/$\overline{W}$ lines return to the bus. Because it is too late in cycle #5 to access memory, this cycle is dead and used for synchronization. Program execution resumes in cycle #6.

- **Valid Memory Address (VMA)**

This output indicates to peripheral devices that there is a valid address on the address bus. In normal operation, this signal should be utilized for enabling peripheral interfaces such as the PIA and ACIA. This signal is not three-state. One standard TTL load and 90pF may be directly driven by this active "High" signal.

- **Halt ($\overline{HALT}$)**

When this input is in the "Low" state, all activity in the machine will be halted. This input is level sensitive.

The $\overline{HALT}$ line provides an input to the MPU to allow control or program execution by an outside source. If $\overline{HALT}$ is "High", the MPU will execute the instructions; if it is "Low", the MPU will go to a halted or idle mode. A response signal, Bus Available (BA) provides an indication of the current MPU status. When BA is "Low", the MPU is in the process of executing the control program; if BA is "High", the MPU has halted and all internal activity has stopped.

When BA is "High", the Address Bus, Data Bus, and R/$\overline{W}$ line will be in a high impedance state, effectively removing the MPU from the system bus. VMA is forced "Low" so that the floating system bus will not activate any device on the bus that is enabled by VMA.

While the MPU is halted, all program activity is stopped, and if either an $\overline{NMI}$ or $\overline{IRQ}$ interrupt occurs, it will be latched into the MPU and acted on as soon as the MPU is taken out of the halted mode. If a $\overline{RES}$ command occurs while the MPU is halted, the following states occur: VMA = "Low", BA = "Low", Data Bus = high impedance, R/$\overline{W}$ = "High" (read state), and the Address Bus will contain address FFFE as long as $\overline{RES}$ is "Low". As soon as the $\overline{RES}$ line goes "High", the MPU will go to locations FFFE and FFFF for the address of the reset routine.

Fig. 18 shows the timing relationships involved when halting the MPU. The instruction illustrated is a one byte, 2 cycle instruction such as CLRA. When $\overline{HALT}$ goes "Low", the MPU will halt after completing execution of the current instruction. The transition of $\overline{HALT}$ must occur t_{PCS} before the trailing edge of ϕ_1 of the last cycle of an instruction (point A of Fig. 18). $\overline{HALT}$ must not go "Low" any time later than the minimum t_{PCS} specified.

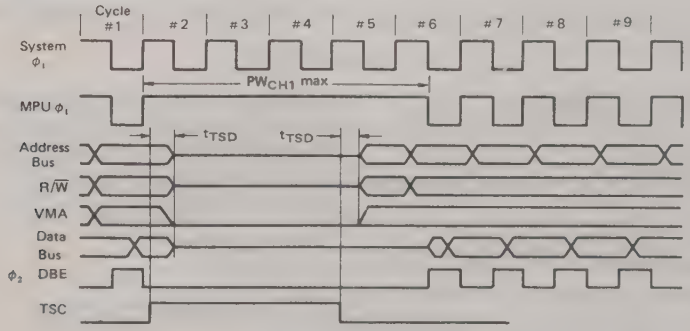


Figure 16 TSC Control Timing

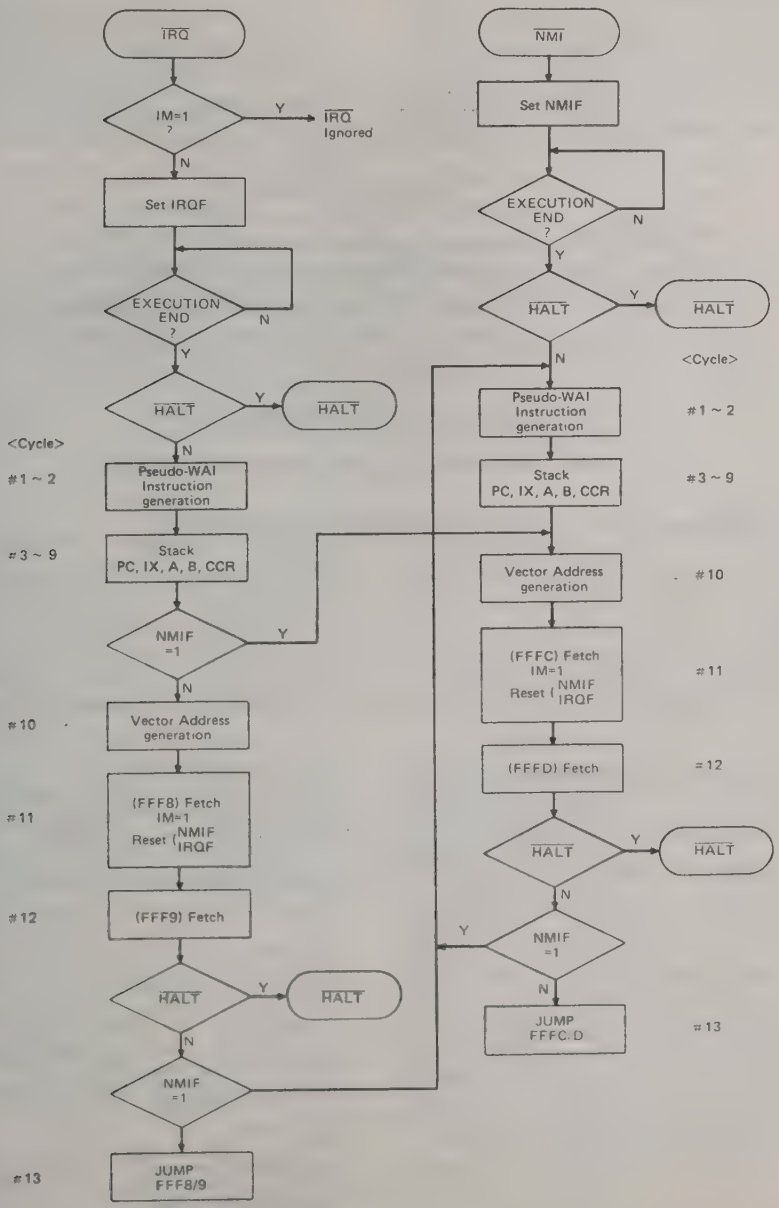
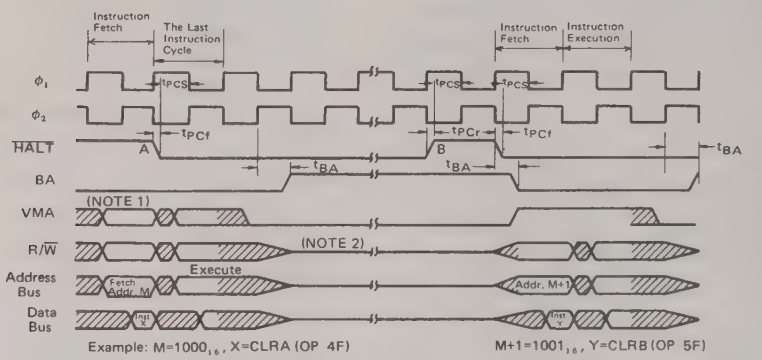


Figure 17 MPU Interrupt Flow Chart



(NOTE) 1. Oblique lines indicate indeterminate range of data.
2. Midrange waveform indicates high impedance,state.

Figure 18 $\overline{\text{HALT}}$ and Single Instruction Execution for System Dubug

Table 2 Operation States of MPU and Signal Outputs (Except the Execution of Instruction)

Signals	Halt state	Reset state	Halt and Reset state	WAI state	TSC state
BA	"H"	"L"	"L"	"H"	"L"
VMA	"L"	"L"	"L"	"L"	"L"
R/$\overline{W}$	"T"	"H"	"H"	"T"	"T"
$A_0 \sim A_{15}$	"T"	$(FFFE)_{16}$	$(FFFE)_{16}$	"T"	"T"
$D_0 \sim D_7$	"T"	"T"	"T"	"T"	–

"T" indicates high impedance state.

The fetch of the OP code by the MPU is the first cycle of the instruction. If $\overline{\text{HALT}}$ had not been "Low" at Point A but went "Low" during ϕ_2 of the cycle, the MPU would have halted after completion of the following instruciton. BA will go "High" by time t_{BA} (bus available delay time) after the last instruction cycle. At this point in time, VMA is "Low" and R/$\overline{W}$, Address Bus, and the Data Bus are in the high impedance state.

To debug programs it is advantageous to step through programs instruction by instruction. To do this, $\overline{\text{HALT}}$ must be brought "High" for one MPU cycle and then returned "Low" as shown at point B of Fig. 18. Again, the transitions of $\overline{\text{HALT}}$ must occur t_{PCS} before the trailing edge of ϕ_1. BA will go "Low" at t_{BA} after the leading edge of the next ϕ_1, indicating that the Address Bus, Data Bus, VMA and R/$\overline{W}$ lines are back on the bus. A single byte, 2 cycle instruction such as LSR is used for this example also. During the first cycle, the instruction Y is fetched from address M+1. BA returns "High" at t_{BA} on the last cycle of the instruction indicating the MPU is off the bus, if instruction Y had been three cycles, the width of the BA "Low" time would have been increased by one cycle.

Table 2 shows the relation between the state of MPU and signal outputs.

■ MPU INSTRUCTION SET

This Section will provide a brief introduction and discuss their use in developing HD6800 MPU control programs. The HD6800 MPU has a set of 72 different executable source instructions. Included are binary and decimal arithmetic, logical, shift, rotate, load, store, conditional or unconditional branch, interrupt and stack manipulation instructions.

Each of the 72 executable instructions of the source language assembles into 1 to 3 bytes of machine code. The number of bytes depends on the particular instruction and on the addressing mode. (The addressing modes which are available for use with the various executive instructions are discussed later.)

The coding of the first (or only) byte corresponding to an executable instruction is sufficient to identify the instruction and the addressing mode. The hexadecimal equivalents of the binary codes, which result from the translation of the 72 instructions in all valid modes of addressing, are shown in Table 3. There are 197 valid machine codes, 59 of the 256 possible codes being unassigned.

When an instruction translates into two or three bytes of code, the second byte, or second and third bytes contain(s) an operand, an address, or information from which an address is obtained during execution.

Microprocessor instructions are often devided into three general classifications; (1) memory reference, so called because they operate on specific memory locations; (2) operating instructions that function without needing a memory reference; (3) I/O instructions for transferring data between the microprocessor and peripheral devices.

In many instances, the HD6800 MPU performs the same operation on both its internal accumulators and the external memory locations. In addition, the HD6800 MPU allow the MPU to treat peripheral devices exactly like other memory locations, hence, no I/O instructions as such are required. Because of these features, other classifications are more suitable for introducing the HD6800's instruction set: (1) Accumulator and memory operations; (2) Program control operations; (3) Condition Code Register operations.

For Accumulator and Memory Operations, refer to Table 4.

Table 3 Hexadecimal Values of Machine Codes

MSB \ LSB	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	•	NOP (IMP)	•	•	•	•	TAP (IMP)	TPA (IMP)	INX (IMP)	DEX (IMP)	CLV (IMP)	SEV (IMP)	CLC (IMP)	SEC (IMP)	CLI (IMP)	SEI (IMP)
1	SBA (A, B)	CBA (A, B)	•	•	•	•	TAB (IMP)	TBA (IMP)	•	DAA (IMP)	•	ABA (IMP)	•	•	•	•
2	BRA (REL)	•	BHI (REL)	BLS (REL)	BCC (REL)	BCS (REL)	BNE (REL)	BEQ (REL)	BVC (REL)	BVS (REL)	BPL (REL)	BMI (REL)	BGE (REL)	BLT (REL)	BGT (REL)	BLE (REL)
3	TSX (IMP)	INS (IMP)	PUL (A)	PUL (B)	DES (IMP)	TXS (IMP)	PSH (A)	PSH (B)	•	RTS (IMP)	•	RTI (IMP)	•	•	WAI (IMP)	SWI (IMP)
4	NEG (A)	•	•	COM (A)	LSR (A)	•	ROR (A)	ASR (A)	ASL (A)	ROL (A)	DEC (A)	•	INC (A)	TST (A)	•	CLR (A)
5	NEG (B)	•	•	COM (B)	LSR (B)	•	ROR (B)	ASR (B)	ASL (B)	ROL (B)	DEC (B)	•	INC (B)	TST (B)	•	CLR (B)
6	NEG (IND)	•	•	COM (IND)	LSR (IND)	•	ROR (IND)	ASR (IND)	ASL (IND)	ROL (IND)	DEC (IND)	•	INC (IND)	TST (IND)	JMP (IND)	CLR (IND)
7	NEG (EXT)	•	•	COM (EXT)	LSR (EXT)	•	ROR (EXT)	ASR (EXT)	ASL (EXT)	ROL (EXT)	DEC (EXT)	•	INC (EXT)	TST (EXT)	JMP (EXT)	CLR (EXT)
8	SUB (IMM)(A)	CMP (IMM)(A)	SBC (IMM)(A)	•	AND (IMM)(A)	BIT (IMM)(A)	LDA (IMM)(A)	•	EOR (IMM)(A)	ADC (IMM)(A)	ORA (IMM)(A)	ADD (IMM)(A)	CPX (IMM)(A)	BSR (REL)	LDS (IMM)	•
9	SUB (DIR)(A)	CMP (DIR)(A)	SBC (DIR)(A)	•	AND (DIR)(A)	BIT (DIR)(A)	LDA (DIR)(A)	STA (DIR)(A)	EOR (DIR)(A)	ADC (DIR)(A)	ORA (DIR)(A)	ADD (DIR)(A)	CPX (DIR)(A)	•	LDS (DIR)	STS (DIR)
A	SUB (IND)(A)	CMP (IND)(A)	SBC (IND)(A)	•	AND (IND)(A)	BIT (IND)(A)	LDA (IND)(A)	STA (IND)(A)	EOR (IND)(A)	ADC (IND)(A)	ORA (IND)(A)	ADD (IND)(A)	CPX (IND)(A)	JSR (IND)	LDS (IND)	STS (IND)
B	SUB (EXT)(A)	CMP (EXT)(A)	SBC (EXT)(A)	•	AND (EXT)(A)	BIT (EXT)(A)	LDA (EXT)(A)	STA (EXT)(A)	EOR (EXT)(A)	ADC (EXT)(A)	ORA (EXT)(A)	ADD (EXT)(A)	CPX (EXT)(A)	JSR (EXT)	LDS (EXT)	STS (EXT)
C	SUB (IMM)(B)	CMP (IMM)(B)	SBC (IMM)(B)	•	AND (IMM)(B)	BIT (IMM)(B)	LDA (IMM)(B)	•	EOR (IMM)(B)	ADC (IMM)(B)	ORA (IMM)(B)	ADD (IMM)(B)	•	•	LDX (IMM)	•
D	SUB (DIR)(B)	CMP (DIR)(B)	SBC (DIR)(B)	•	AND (DIR)(B)	BIT (DIR)(B)	LDA (DIR)(B)	STA (DIR)(B)	EOR (DIR)(B)	ADC (DIR)(B)	ORA (DIR)(B)	ADD (DIR)(B)	•	•	LDX (DIR)(B)	STX (DIR)(B)
E	SUB (IND)(B)	CMP (IND)(B)	SBC (IND)(B)	•	AND (IND)(B)	BIT (IND)(B)	LDA (IND)(B)	STA (IND)(B)	EOR (IND)(B)	ADC (IND)(B)	ORA (IND)(B)	ADD (IND)(B)	•	•	LDX (IND)	STX (IND)
F	SUB (EXT)(B)	CMP (EXT)(B)	SBC (EXT)(B)	•	AND (EXT)(B)	BIT (EXT)(B)	LDA (EXT)(B)	STA (EXT)(B)	EOR (EXT)(B)	ADC (EXT)(B)	ORA (EXT)(B)	ADD (EXT)(B)	•	•	LDX (EXT)	STX (EXT)

DIR = Direct Addressing Mode
EXT = Extended Addressing Mode
IMM = Immediate Addressing Mode
IND = Index Addressing Mode
IMP = Implied Addressing Mode
REL = Relative Addressing Mode
A = Accumulator A
B = Accumulator B

Table 4 Accumulator and Memory Operations

Operation	Mnemonic	Addressing Modes: IMMED OP ~	IMMED #	DIRECT OP ~	DIRECT #	INDEX OP ~	INDEX #	EXTND OP ~	EXTND #	IMPLIED OP ~	IMPLIED #	Boolean/ Arithmetic Operation	Cond. Code Reg. 5 H	4 I	3 N	2 Z	1 V	0 C
Add	ADDA	8B	2	9B	3	AB	5	BB	4			A + M → A	↕	•	↕	↕	↕	↕
	ADDB	CB	2	DB	3	EB	5	FB	4			B + M → B	↕	•	↕	↕	↕	↕
Add Acmltrs	ABA									1B	2	A + B → A	↕	•	↕	↕	↕	↕
Add with Carry	ADCA	89	2	99	3	A9	5	B9	4			A + M + C → A	↕	•	↕	↕	↕	↕
	ADCB	C9	2	D9	3	E9	5	F9	4			B + M + C → B	↕	•	↕	↕	↕	↕
And	ANDA	84	2	94	3	A4	5	B4	4			A • M → A	•	•	↕	↕	R	•
	ANDB	C4	2	D4	3	E4	5	F4	4			B • M → B	•	•	↕	↕	R	•
Bit Test	BITA	85	2	95	3	A5	5	B5	4			A • M	•	•	↕	↕	R	•
	BITB	C5	2	D5	3	E5	5	F5	4			B • M	•	•	↕	↕	R	•
Clear	CLR					6F	7	7F	6			00 → M	•	•	R	S	R	R
	CLRA									4F	2	00 → A	•	•	R	S	R	R
	CLRB									5F	2	00 → B	•	•	R	S	R	R
Compare	CMPA	81	2	91	3	A1	5	B1	4			A − M	•	•	↕	↕	↕	↕
	CMPB	C1	2	D1	3	E1	5	F1	4			B − M	•	•	↕	↕	↕	↕
Compare Acmltrs	CBA									11	2	A − B	•	•	↕	↕	↕	↕
Complement, 1's	COM					63	7	73	6			M̄ → M	•	•	↕	↕	R	S
	COMA									43	2	Ā → A	•	•	↕	↕	R	S
	COMB									53	2	B̄ → B	•	•	↕	↕	R	S
Complement, 2's (Negate)	NEG					60	7	70	6			00 − M → M	•	•	↕	↕	①	②
	NEGA									40	2	00 − A → A	•	•	↕	↕	①	②
	NEGB									50	2	00 − B → B	•	•	↕	↕	①	②
Decimal Adjust, A	DAA									19	2	Converts Binary Add of BCD Characters into BCD Format	•	•	↕	↕	↕	③
Decrement	DEC					6A	7	7A	6			M − 1 → M	•	•	↕	↕	④	•
	DECA									4A	2	A − 1 → A	•	•	↕	↕	④	•
	DECB									5A	2	B − 1 → B	•	•	↕	↕	④	•
Exclusive OR	EORA	88	2	98	3	A8	5	B8	4			A ⊕ M → A	•	•	↕	↕	R	•
	EORB	C8	2	D8	3	E8	5	F8	4			B ⊕ M → B	•	•	↕	↕	R	•
Increment	INC					6C	7	7C	6			M + 1 → M	•	•	↕	↕	⑤	•
	INCA									4C	2	A + 1 → A	•	•	↕	↕	⑤	•
	INCB									5C	2	B + 1 → B	•	•	↕	↕	⑤	•
Load Acmltr	LDAA	86	2	96	3	A6	5	B6	4			M → A	•	•	↕	↕	R	•
	LDAB	C6	2	D6	3	E6	5	F6	4			M → B	•	•	↕	↕	R	•
Or, Inclusive	ORAA	8A	2	9A	3	AA	5	BA	4			A + M → A	•	•	↕	↕	R	•
	ORAB	CA	2	DA	3	EA	5	FA	4			B + M → B	•	•	↕	↕	R	•
Push Data	PSHA									36	4	A → Msp, SP − 1 → SP	•	•	•	•	•	•
	PSHB									37	4	B → Msp, SP − 1 → SP	•	•	•	•	•	•
Pull Data	PULA									32	4	SP + 1 → SP, Msp → A	•	•	•	•	•	•
	PULB									33	4	SP + 1 → SP, Msp → B	•	•	•	•	•	•
Rotate Left	ROL					69	7	79	6			M	•	•	↕	↕	⑥	↕
	ROLA									49	2	A	•	•	↕	↕	⑥	↕
	ROLB									59	2	B — C b7 ← b0	•	•	↕	↕	⑥	↕
Rotate Right	ROR					66	7	76	6			M	•	•	↕	↕	⑥	↕
	RORA									46	2	A	•	•	↕	↕	⑥	↕
	RORB									56	2	B — C b7 → b0	•	•	↕	↕	⑥	↕
Shift Left, Arithmetic	ASL					68	7	78	6			M	•	•	↕	↕	⑥	↕
	ASLA									48	2	A	•	•	↕	↕	⑥	↕
	ASLB									58	2	B — C b7 b0	•	•	↕	↕	⑥	↕
Shift Right, Arithmetic	ASR					67	7	77	6			M	•	•	↕	↕	⑥	↕
	ASRA									47	2	A	•	•	↕	↕	⑥	↕
	ASRB									57	2	B — b7 b0 C	•	•	↕	↕	⑥	↕
Shift Right, Logic	LSR					64	7	74	6			M	•	•	R	↕	⑥	↕
	LSRA									44	2	A	•	•	R	↕	⑥	↕
	LSRB									54	2	B — b7 b0 C	•	•	R	↕	⑥	↕
Store Acmltr	STAA			97	4	A7	6	B7	5			A → M	•	•	↕	↕	R	•
	STAB			D7	4	E7	6	F7	5			B → M	•	•	↕	↕	R	•
Subtract	SUBA	80	2	90	3	A0	5	B0	4			A − M → A	•	•	↕	↕	↕	↕
	SUBB	C0	2	D0	3	E0	5	F0	4			B − M → B	•	•	↕	↕	↕	↕
Subtract Acmltrs	SBA									10	2	A − B → A	•	•	↕	↕	↕	↕
Subtr with Carry	SBCA	82	2	92	3	A2	5	B2	4			A − M − C → A	•	•	↕	↕	↕	↕
	SBCB	C2	2	D2	3	E2	5	F2	4			B − M − C → B	•	•	↕	↕	↕	↕
Transfer Acmltrs	TAB									16	2	A → B	•	•	↕	↕	R	•
	TBA									17	2	B → A	•	•	↕	↕	R	•
Test Zero or Minus	TST					6D	7	7D	6			M − 00	•	•	↕	↕	R	R
	TSTA									4D	2	A − 00	•	•	↕	↕	R	R
	TSTB									5D	2	B − 00	•	•	↕	↕	R	R

LEGEND:

OP Operation Code (Hexadecimal)
~ Number of MPU Cycles
# Number of Program Bytes
+ Arithmetic Plus
− Arithmetic Minus
• Boolean AND
Msp Contents of memory location pointed to be Stack Pointer
+ Boolean Inclusive OR
⊕ Boolean Exclusive OR
M̄ Complement of M
→ Transfer Into
0 Bit = Zero
00 Byte = Zero

CONDITION CODE SYMBOLS:

H Half-carry from bit 3
I Interrupt mask
N Negative (sign bit)
Z Zero (byte)
V Overflow, 2's complement
C Carry from bit 7
R Reset Always
S Set Always
↕ Test and set if true, cleared otherwise
• Not Affected

(Note) Accumulator addressing mode instructions are included in the column for IMPLIED addressing

CONDITION CODE REGISTER NOTES:
(Bit set if test is true and cleared otherwise)

① (Bit V) Test: Result = 10000000?
② (Bit C) Test: Result = 00000000?
③ (Bit C) Test: Decimal value of most significant BCD Character greater than nine? (Not cleared if previously set.)
④ (Bit V) Test: Operand = 10000000 prior to execution?
⑤ (Bit V) Test: Operand = 01111111 prior to execution?
⑥ (Bit V) Test: Set equal to result of N⊕C after shift has occurred.

■ PROGRAM CONTROL OPERATIONS

Program Control operation can be subdivided into two categories: (1) Index Register/Stack Pointer instructions: (2) Jump and Branch of operations.

● Index Register/Stack Pointer Operations

The instructions for direct operation on the MPU's Index Register and Stack Pointer are summarized in Table 5. Decrement (DEX, DES), increment (INX, INS), load (LDX, LDS), and store (STX, STS) instructions are provided for both. The Compare instruction, CPX, can be used to compare the Index Register to a 16-bit value and update the Condition Code Register accordingly.

The TSX instruction causes the Index Register to be loaded with the address of the last data byte put onto the "stack". The TXS instruction loads the Stack Pointer with a value equal to one less than the current contents of the Index Register. This causes the next byte to be pulled from the "stack" to come from the location indicated by the Index Register. The utility of these two instructions can be clarified by describing the "stack" concept relative to the HMCS 6800 system.

The "stack" can be thought of as a sequential list of data stored in the MPU's read/write memory. The Stack Pointer contains a 16-bit memory address that is used to access the list from one end on a last-in-first-out (LIFO) basis in contrast to the random access mode used by the MPU's other addressing modes.

The HD6800 MPU instruction set and interrupt structure allow extensive use of the stack concept for efficient handling of data movement, subroutines and interrupts. The instructions can be used to establish one or more "stacks" anywhere in read/write memory. Stack length is limited only by the amount of memory that is made available.

Operation of the Stack Pointer with the Push and Pull instructions is illustrated in Figs. 19 and 20. The Push instruction (PSHA) causes the contents of the indicated accumulator (A in this example) to be stored in memory at the location indicated by the Stack Pointer. The Stack Pointer is automatically decremented by one following the storage operation and is "pointing" to the next empty stack location.

The Pull instruction (PULA or PULB) causes the last byte stacked to be loaded into the appropriate accumulator. The Stack Pointer is automatically incremented by one just prior to the data transfer so that it will point to the last byte stacked rather than the next empty location. Note that the PULL instruction does not "remove" the data from memory; in the example, 1A is still in location (m+1) following execution of PULA. A subsequent PUSH instruction would overwrite than location with the new "pushed" data.

Execution of the Branch to Subroutine (BSR) and Jump to Subroutine (JSR) instructions cause a return address to be save on the stack as shown in Figs. 21 through 23. The stack is decremented after each byte of the return address is pushed onto the stack. For both of the these instructions, the return address is the memory location following the bytes of code that correspond to the BSR and JSR instruction. The code required for BSR or JSR may be either two or three bytes, depending on whether the JSR is in the indexed (two bytes) or the extended (three bytes) addressing mode. Before it is stacked, the Program Counter is automatically incremented the correct number of times to be pointing at the location of the next instruction. The Return from Subroutine instruction, RTS, causes the return address to be retrieved and loaded into the Program Counter as shown in Fig. 24.

There are several operations that cause the status of the MPU to be saved on the stack. The Software Interrupt (SWI) and Wait for Interrupt (WAI) instructions as well as the maskable ($\overline{IRQ}$) and non-maskable ($\overline{NMI}$) hardware interrupts all cause the MPU's internal registers (except for the Stack Pointer itself) to be stacked as shown in Fig. 25. MPU status is restored by the Return from interrupt, RTI, as shown in Fig. 26.

Table 5 Index Register and Stack Pointer Instructions

Operation	Mnemonic	Addressing Modes														Boolean/ Arithmetic Operation	Cond. Code Reg.					
		IMMED			DIRECT			INDEX			EXTND			IMPLIED				5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#		H	I	N	Z	V	C
Compare Index Reg	CPX	8C	3	3	9C	4	2	AC	6	2	BC	5	3				$(X_H) - (M), (X_L) - (M+1)$	•	•	①	↕	②	•
Decrement Index Reg	DEX													09	4	1	$X - 1 \rightarrow X$	•	•	•	↕	•	•
Decrement Stack Pntr	DES													34	4	1	$SP - 1 \rightarrow SP$	•	•	•	•	•	•
Increment Index Reg	INX													08	4	1	$X + 1 \rightarrow X$	•	•	•	↕	•	•
Increment Stack Pntr	INS													31	4	1	$SP + 1 \rightarrow SP$	•	•	•	•	•	•
Load Index Reg	LDX	CE	3	3	DE	4	2	EE	6	2	FE	5	3				$M \rightarrow X_H, (M+1) \rightarrow X_L$	•	•	③	↕	R	•
Load Stack Pntr	LDS	8E	3	3	9E	4	2	AE	6	2	BE	5	3				$M \rightarrow SP_H, (M+1) \rightarrow SP_L$	•	•	③	↕	R	•
Store Index Reg	STX				DF	5	2	EF	7	2	FF	6	3				$X_H \rightarrow M, X_L \rightarrow (M+1)$	•	•	③	↕	R	•
Store Stack Pntr	STS				9F	5	2	AF	7	2	BF	6	3				$SP_H \rightarrow M, SP_L \rightarrow (M+1)$	•	•	③	↕	R	•
Index Reg → Stack Pntr	TXS													35	4	1	$X - 1 \rightarrow SP$	•	•	•	•	•	•
Stack Pntr → Index Reg	TSX													30	4	1	$SP + 1 \rightarrow X$	•	•	•	•	•	•

① (Bit N) Test: Sign bit of most significant (MS) byte of result = 1?
② (Bit V) Test: 2's complement overflow from subtraction of ms bytes?
③ (Bit N) Test: Result less than zero? (Bit 15 = 1)

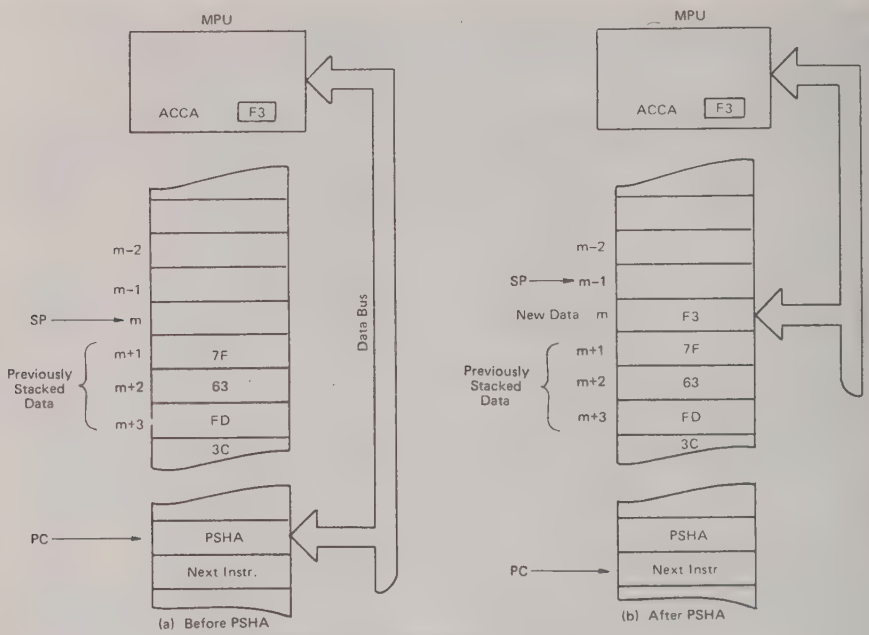


Figure 19 Stack Operation (Push Instruction)

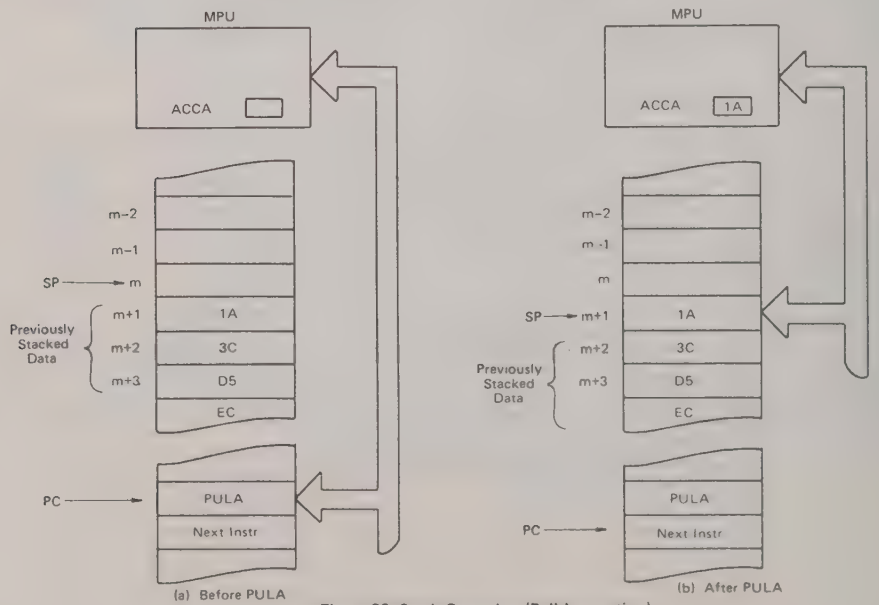


Figure 20 Stack Operation (Pull Instruction)

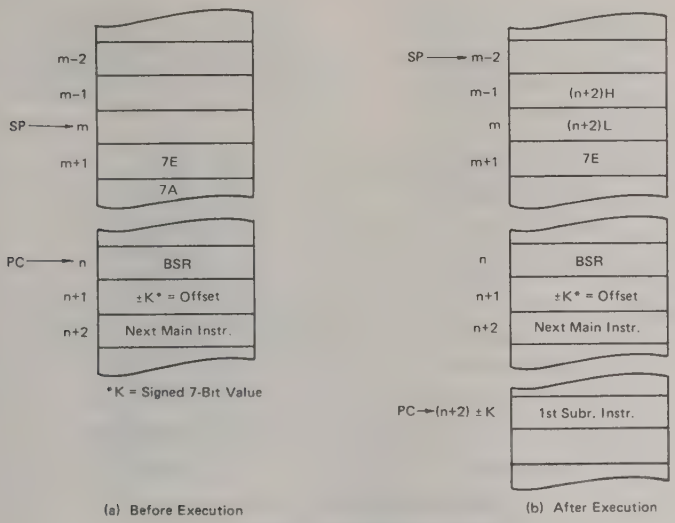


Figure 21 Program Flow for BSR

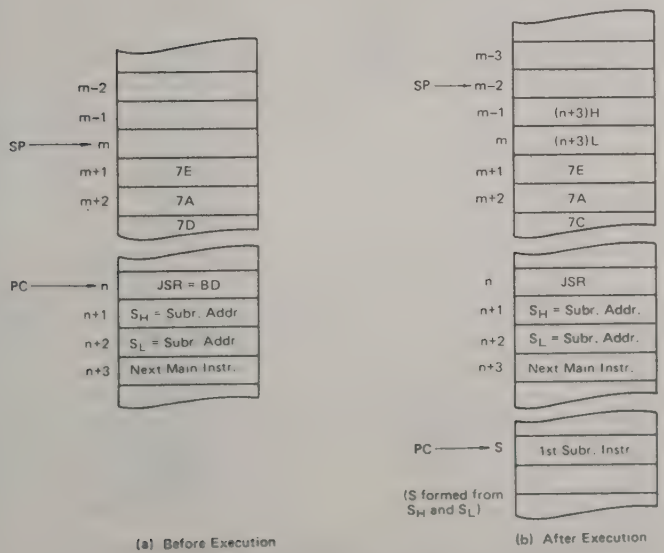


Figure 22 Program Flow for JSR (Extended)

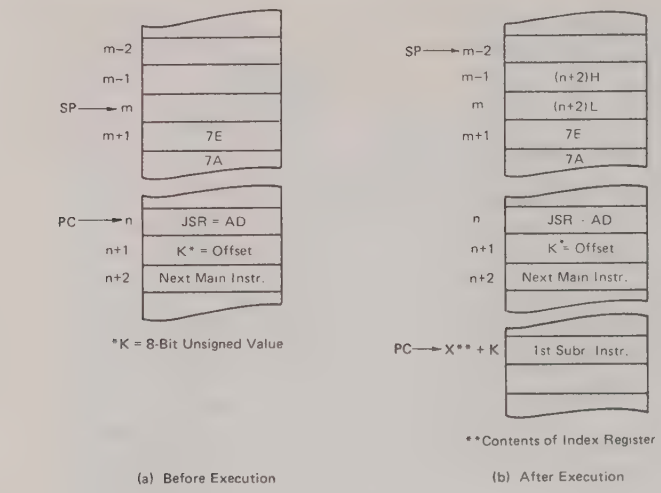


Figure 23 Program Flow for JSR (Indexed)

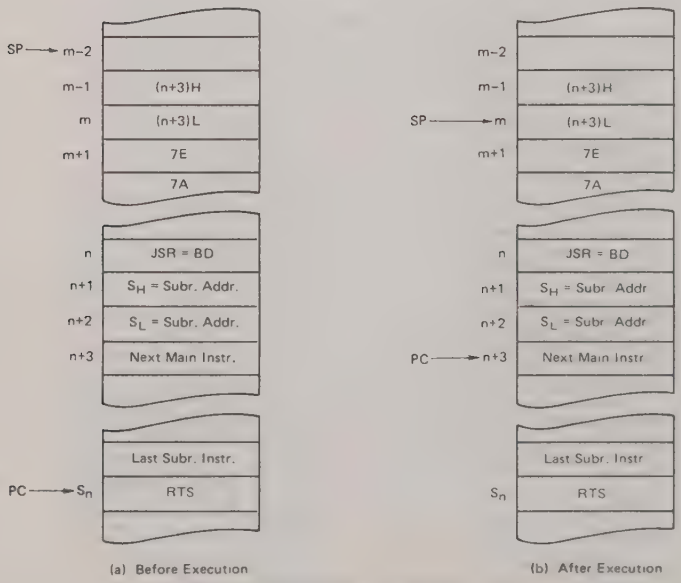


Figure 24 Program Flow for RTS

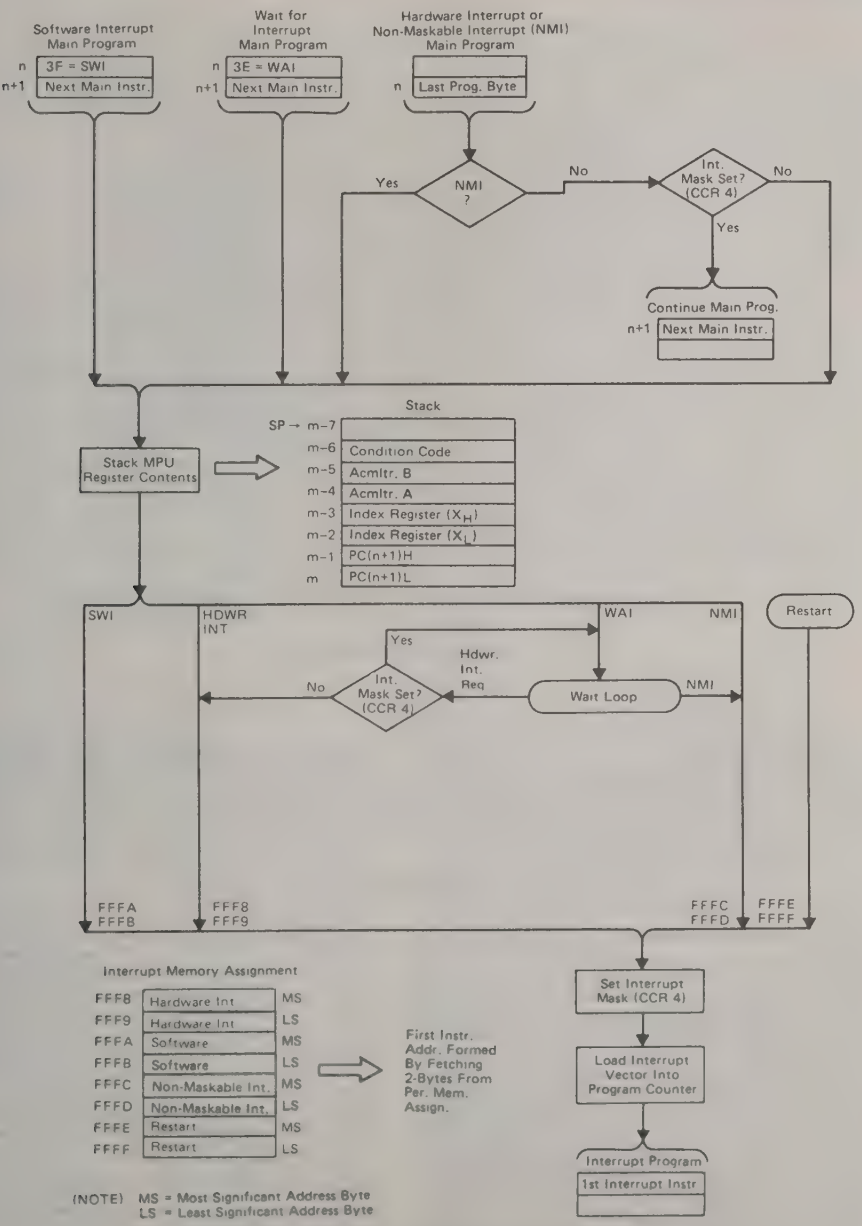


Figure 25 Program Flow for Interrupts

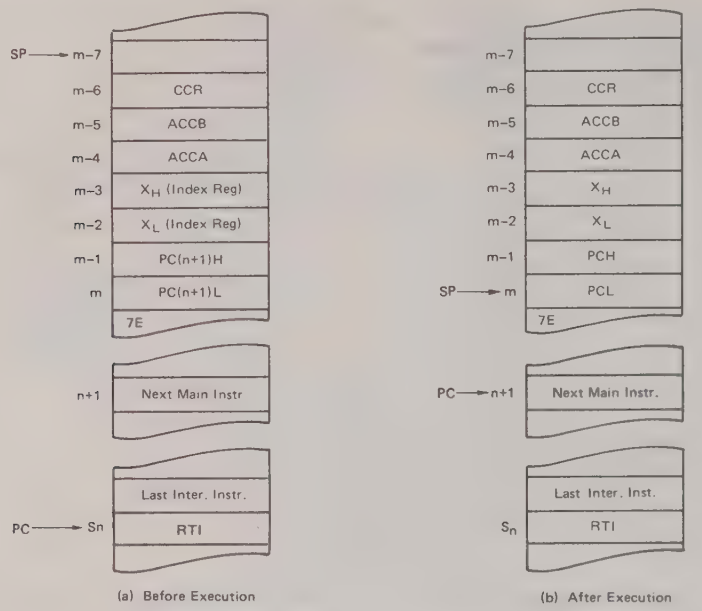


Figure 26 Program Flow for RTI

- **Jump and Branch Operation**

The Jump and Branch instructions are summarized in Table 6. These instructions are used to control the transfer of operation from one point to another in the control program.

The No Operation instruction, NOP, while included here, is a jump operation in a very limited sense. Its only effect is to increment the Program Counter by one. It is useful during program development as a "stand-in" for some other instruction that is to be determined during debug. It is also used for equalizing the execution time through alternate paths in a control program.

Execution of the Jump Instruction, JMP, and Branch Always, BRA, affects program flow as shown in Fig. 27. When the MPU encounters the Jump (Index) instruction, it adds the offset to the value in the Index Register and uses the result as the address of the next instruction to be executed. In the extended addressing mode, the address of the next instruction to be executed is fetched from the two locations immediately following the JMP instruction. The Branch Always (BRA) instruction is similar to the JMP (extended) instruction except that the relative addressing mode applies and the branch is limited to the range within –125 or +127 bytes of the branch instruction itself. The opcode for the BRA instruction requires one less byte than JMP (extended) but takes one more cycle to execute.

The effect on program flow for the Jump to Subroutine (JSR) and Branch to Subroutine (BSR) is shown in Figs. 21 through 23. Note that the Program Counter is properly incremented to be pointing at the correct return address before it is stacked. Operation of the Branch to Subroutine and Jump to Subroutine (extended) instruction is similar except for the range. The BSR instruction requires less opcode than JSR (2 bytes versus 3 bytes) and also executes one cycle faster than JSR. The Return from Subroutine, RTS, is used at the end of a subroutine to return to the main program as indicated in Fig. 24.

The effect of executing the Software Interrupt, SWI, and the Wait for Interrupt, WAI, and their relationship to the hardware interrupts is shown in Fig. 25. SWI causes the MPU contents to be stacked and then fetches the starting address of the interrupt routine from the memory locations that respond to the addresses FFFA and FFFB. Note that as in the case of the subroutine instructions, the Program Counter is incremented to point at the correct return address before being stacked. The Return from Interrupt instruction, RTI, (Fig. 26) is used at the end of an interrupt routine to restore control to the main program. The SWI instruction is useful for inserting break points in the control program, that is, it can be used to stop operation and put the MPU registers in memory where they can be examined. The WAI instruction is used to decrease the time required to service a hardware interrupt; it stacks the MPU contents and then waits for the interrupt to occur, effectively removing the stacking time from a hardware interrupt sequence.

Table 6 JUMP/BRANCH Instruction

Operation	Mnemonic	Addressing Modes												Branch Test	Cond. Code Reg.					
		RELATIVE			INDEX			EXTND			IMPLIED				5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#		H	I	N	Z	V	C
Branch Always	BRA	20	4	2										None	•	•	•	•	•	•
Branch If Carry Clear	BCC	24	4	2										C = 0	•	•	•	•	•	•
Branch If Carry Set	BCS	25	4	2										C = 1	•	•	•	•	•	•
Branch If = Zero	BEQ	27	4	2										Z = 1	•	•	•	•	•	•
Branch If ≧ Zero	BGE	2C	4	2										N ⊕ V = 0	•	•	•	•	•	•
Branch If > Zero	BGT	2E	4	2										Z + (N ⊕ V) = 0	•	•	•	•	•	•
Branch If Higher	BHI	22	4	2										C + Z = 0	•	•	•	•	•	•
Branch If ≦ Zero	BLE	2F	4	2										Z + (N ⊕ V) = 1	•	•	•	•	•	•
Branch If Lower Or Same	BLS	23	4	2										C + Z = 1	•	•	•	•	•	•
Branch If < Zero	BLT	2D	4	2										N ⊕ V = 1	•	•	•	•	•	•
Branch If Minus	BMI	2B	4	2										N = 1	•	•	•	•	•	•
Branch If Not Equal Zero	BNE	26	4	2										Z = 0	•	•	•	•	•	•
Branch If Overflow Clear	BVC	28	4	2										V = 0	•	•	•	•	•	•
Branch If Overflow Set	BVS	29	4	2										V = 1	•	•	•	•	•	•
Branch If Plus	BPL	2A	4	2										N = 0	•	•	•	•	•	•
Branch To Subroutine	BSR	8D	8	2											•	•	•	•	•	•
Jump	JMP				6E	4	2	7E	3	3					•	•	•	•	•	•
Jump To Subroutine	JSR				AD	8	2	BD	9	3					•	•	•	•	•	•
No Operation	NOP										01	2	1	Advances Prog Cntr Only	•	•	•	•	•	•
Return From Interrupt	RTI										3B	10	1		①					
Return From Subroutine	RTS										39	5	1		•	•	•	•	•	•
Software Interrupt	SWI										3F	12	1		•	S	•	•	•	•
Wait for Interrupt	WAI										3E	9	1		•	②	•	•	•	•

① (All) Load Condition Code Register from Stack. (See Special Operations)
② (Bit I) Set when interrupt occurs. If previously set, a Non-Maskable interrupt is required to exit the wait state.

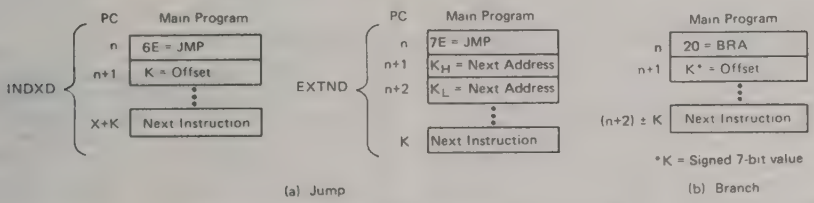


Figure 27 Program Flow for JUMP/BRANCH Instructions

BMI : N = 1 ; BEQ : Z = 1 ;
BPL : N = 0 ; BNE : Z = 0 ;

BVC : V = 0 ; BCC : C = 0 ;
BVS : V = 1 ; BCS : C = 1 ;

BHI : C + Z = 0 ; BLT : N ⊕ V = 1 ;
BLS : C + Z = 1 ; BGE : N ⊕ V = 0 ;

BLE : Z + (N ⊕ V) = 1 ;
BGT : Z + (N ⊕ V) = 0 ;

Figure 28 Conditional Branch Instructions

The conditional branch instructions, Fig. 28, consists of seven pairs of complementary instructions. They are used to test the results of the preceding operation and either continue with the next instruction in sequence (test fails) or cause a branch to another point in the program (test succeeds).

Four of the pairs are used for simple tests of status bits N, Z, V, and C:

1. Branch on Minus (BMI) and Branch On Plus (BPL) tests the sign bit, N, to determine if the previous result was negative or positive, respectively.
2. Branch On Equal (BEQ) and Branch On Not Equal (BNE) are used to test the zero status bit, Z, to determine whether or not the result of the previous operation was equal to "0". These two instructions are useful following a Compare (CMP) instruction to test for equality between an accumulator and the operand. They are also used following the Bit Test (BIT) to determine whether or not the same bit positions are set in an accumulator and the operand.

3. Branch On Overflow Clear (BVC) and Branch On Overflow Set (BVS) tests the state of the V bit to determine if the previous operation caused an arithmetic overflow.
4. Branch On Carry Clear (BCC) and Branch On Carry Set (BCS) tests the state of the C bit to determine if the previous operation caused a carry to occur. BCC and BCS are useful for testing relative magnitude when the values being tested are regarded as unsigned binary numbers, that is, the values are in the range "00" (lowest) of "FF" (highest). BCC following a comparison (CMP) will cause a branch if the (unsigned) value in the accumulator is higher than or the same as the value of the operand. Conversely, BCS will cause a branch if the accumulator value is lower than the operand.

The Fifth complementary pair, Branch On Higher (BHI) and Branch On Lower or Same (BLS) are in a sense complements to BCC and BCS. BHI tests for both C and Z = "0", if used following a CMP, it will cause a branch if the value in the accumulator is higher than the operand. Conversely, BLS will cause a branch if the unsigned binary value in the accumulator is lower than or the same as the operand.

The remaining two pairs are useful in testing results of operations in which the values are regarded as signed two's complement numbers. This differs from the unsigned binary case in the following sense: In unsigned, the orientation is higher or lower; in signed two's complement, the comparison is between larger or smaller where the range of values is between –128 and +127.

Branch On Less Than Zero (BLT) and Branch On Greater Than Or Equal Zero (BGE) test the status bits for N⊕V = "1" and N ⊕ V = "0", respectively. BLT will always cause a branch following an operation in which two negative numbers were added. In addition, it will cause a branch following a CMP in which the value in the accumulator was negative and the operand was positive. BLT will never cause a branch following a CMP in which the accumulator value was positive and the operand negative. BGE, the complement to BLT, will cause a branch following operations in which two positive values were added or in which the result was "0".

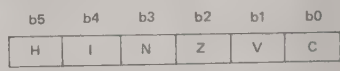
The last pair, Branch On Less Than Or Equal Zero (BLE) and Branch On Greater Than Zero (BGT) test the status bits for Z ⊕ (N + V) = "1" and Z ⊕(N + V) = "0", respectively, The action of BLE is identical to that for BLT except that a branch will also occur if the result of the previous result was "0". Conversely, BGT is similar to BGE except that no branch will occur following a "0" result.

■ CONDITION CODE REGISTER OPERATIONS

The Condition Code Register (CCR) is a 6-bit register within the MPU that is useful in controlling program flow during system operation. The bits are defined in Fig. 29.

The instructions shown in Table 7 are available to the user for direct manipulation of the CCR. In addition, the MPU automatically sets or clears the appropriate status bits as many of the other instructions on the condition code register was indicated as they were introduced.

Systems which require an interrupt window to be opened under program control should use a CLI-NOP-SEI sequence rather than CLI-SEI.



H = Half-carry; set whenever a carry from b3 to b4 of the result is generated by ADD, ABA, ADC; cleared if no b3 to b4 carry; not affected by other instructions.

I = Interrupt Mask; set by hardware of software interrupt or SEI instruction; cleared by CLI instruction. (Normally not used in arithmetic operations.) Restored to a "0" as a result of an RTI instruction if IM stored on the stacked is "0"

N = Negative; set if high order bit (b7) of result is set; cleared otherwise.

Z = Zero; set if result = "0"; cleared otherwise.

V = Overflow; set if there was arithmetic overflow as a result of the operation; cleared otherwise.

C = Carry; set if there was a carry from the most significant bit (b7) of the result; cleared otherwise.

Figure 29 Condition Code Register Bit Definition

■ ADDRESSING MODES

The MPU operates on 8-bit binary numbers presented to it via the Data Bus. A given number (byte) may represent either data or an instruction to be executed, depending on where it is encountered in the control program. The HD6800 MPU has 72 unique instructions, however, it recognizes and takes action on 197 of the 256 possibilities that can occur using an 8-bit word length. This larger number of instructions results from the fact that many of the executive instructions have more than one addressing mode.

Table 7 Condition Code Register Instructions

Operations	Mnemonic	Addressing Mode IMPLIED OP	~	#	Boolean Operation	Cond. Code Reg. 5 H	4 I	3 N	2 Z	1 V	0 C
Clear Carry	CLC	0C	2	1	0 → C	•	•	•	•	•	R
Clear Interrupt Mask	CLI	0E	2	1	0 → I	•	R	•	•	•	•
Clear Overflow	CLV	0A	2	1	0 → V	•	•	•	•	R	•
Set Carry	SEC	0D	2	1	1 → C	•	•	•	•	•	S
Set Interrupt Mask	SEI	0F	2	1	1 → I	•	S	•	•	•	•
Set Overflow	SEV	0B	2	1	1 → V	•	•	•	•	S	•
Acmltr A → CCR	TAP	06	2	1	A → CCR	①					
CCR → Acmltr A	TPA	07	2	1	CCR → A	•	•	•	•	•	•

R = Reset
S = Set
• = Not affected
① (ALL) Set according to the contents of Accumulator A.

These addressing modes refer to the manner in which the program causes the MPU to obtain its instructions and data. The programmer must have a method for addressing the MPU's internal registers and all of the external memory locations.

Selection of the desired addressing mode is made by the user as the source statements are written. Translation into appropriate opcode then depends on the method used. If manual translation is used, the addressing mode is implied in the opcode. For example, the Immediate, Direct, Indexed, and Extended modes may all be used with the ADD instruction. The proper mode is determined by selecting (hexidecimal notation) 8B, 9B, AB, or BB, respectively.

The source statement format includes adequate information for the selection if an assembler program is used to generate the opcode. For instance, the Immediate mode is selected by the Assembler whenever it encounters the "#" symbol in the operand field. Similarly, an "X" in the operand field causes the Indexed mode to be selected. Only the Relative mode applies to the branch instructions, therefore, the mnemonic instruction itself is enough for the Assembler to determine addressing mode.

For the instructions that use both Direct and Extended modes, the Assembler selects the Direct mode if the operand value is in the range 0~255 and Extended otherwise. There are a number of instructions for which the Extended mode is valid but the Direct is not. For these instructions, the Assembler automatically selects the Extended mode even if the operand is in the 0~255 range. The addressing modes are summarized in Fig. 30.

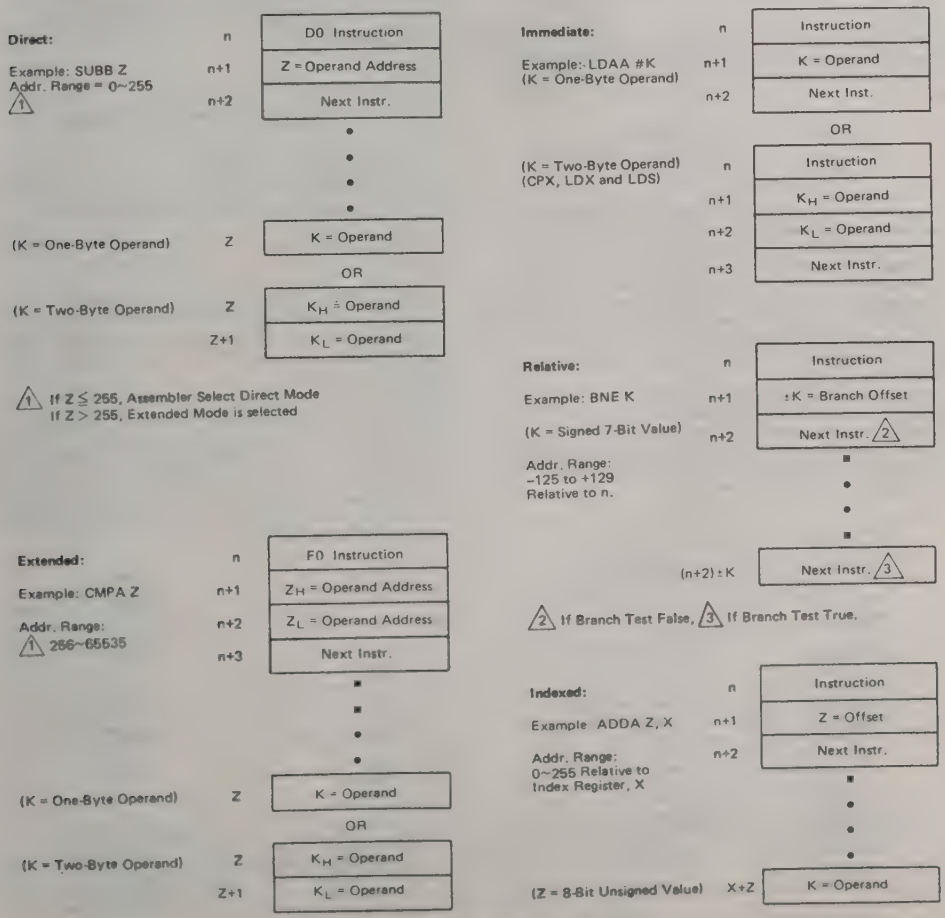


Figure 30 Addressing Mode Summary

- **Implied (Includes "Accumulator Addressing" Mode)**

The successive fields in a statement are normally separated by one or more spaces. An exception to this rule occurs for instructions that use dual addressing in the operand field and for instructions that must distinguish between the two accumulators. In these cases, A and B are "operands" but the space between them and the operator may be omitted. This is commonly done, resulting in apparent four character mnemonics for those instructions.

The addition instruction, ADD, provides an example of dual addressing in the operand fields;

Operator	Operand	Comment
ADDA	MEM12	ADD CONTENTS OF MEM12 TO ACCA
or ADDB	MEM12	ADD CONTENTS OF MEM12 TO ACCB

The example used earlier for the test instruction, TST, also applies to the accumulators and uses the "accumulator addressing mode" to designate which of the two accumulators is being tested:

Operator	Comment
TSTB	TEST CONTENTS OF ACCB
or TSTA	TEST CONTENTS OF ACCA

A number of the instructions either alone or together with an accumulator operand contain all of the address information that is required, that is, "inherent" in the instruction, itself. For instance, the instruction ABA causes the MPU to add the contents of accumulators A and B together and place the result in accumulator A. The instruction INCB, another example of "accumulator addressing", causes the contents of accumulator B to be increased by one. Similarly, INX, increment the Index Register, causes the contents of the Index Register to be increased by one.

Program flow for instructions of this type is illustrated in Figures 31 and 32. In these figures, the general case is shown on the left and a specific example is shown on the right. Numerical examples are in decimal notation. Instructions of this type require only one byte of opcode. Cycle-by-cycle operation of the implied mode is shown in Table 8.

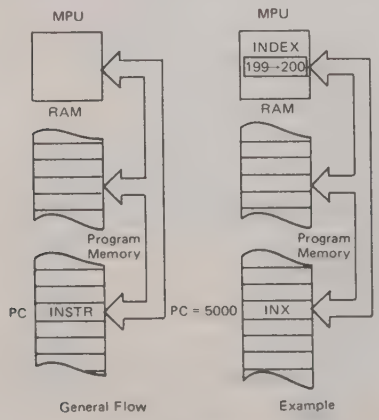


Figure 31 Implied Addressing

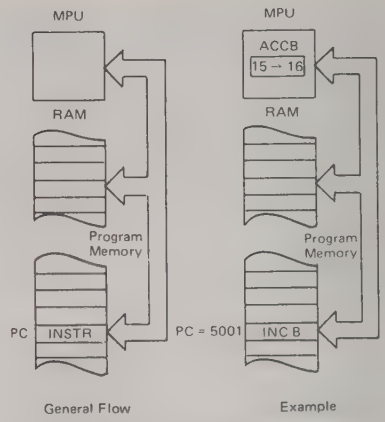


Figure 32 Accumulator Addressing

- **Immediate Addressing Mode**

In the Immediate addressing mode, the operand is the value that is to be operated on. For instance, the instruction

Operator	Operand	Comment
LDAA	#25	LOAD 25 INTO ACCA

causes the MPU to "immediately load accumulator A with the value 25"; no further address reference is required. The Immediate mode is selected by preceding the operand value with the "#" symbol. Program flow for this addressing mode is illustrated in Fig. 33.

The operand format allows either properly defined symbols or numerical values. Except for the instructions CPX, LDX, and LDS, the operand may be any value in the range 0 ~ 255. Since Compare Index Register (CPX), Load Index Register (LDX), Load Stack Pointer (LDS), require 16-bit values, the immediate mode for these three instructions requie two-byte operands.

Table 9 shows the cycle-by-cycle operation for the immediate addressing mode.

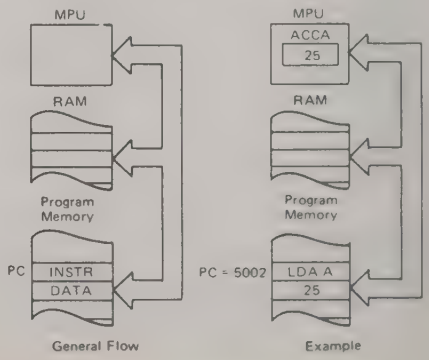


Figure 33 Immediate Addressing Mode

Table 8 Implied Mode Cycle by Cycle Operation

Address Mode and Instructions	Cycle	Cycle #	VMA Line	Address Bus	R/$\overline{W}$ Line	Data Bus
ABA DAA SEC ASL DEC SEI ASR INC SEV CBA LSR TAB CLC NEG TAP CLI NOP TBA CLR ROL TPA CLV ROR TST COM SBA	2	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Op Code of Next Instruction
DES	4	1	1	Op Code Address	1	Op Code
DEX		2	1	Op Code Address + 1	1	Op Code of Next Instruction
INS		3	0	Previous Register Contents	1	Irrelevant Data (NOTE 1)
INX		4	0	New Register Contents	1	Irrelevant Data (NOTE 1)
PSH	4	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Op Code of Next Instruction
		3	1	Stack Pointer	0	Accumulator Data
		4	0	Stack Pointer – 1	1	Accumulator Data
PUL	4	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Op Code of Next Instruction
		3	0	Stack Pointer	1	Irrelevant Data (NOTE 1)
		4	1	Stack Pointer + 1	1	Operand Data from Stack
TSX	4	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Op Code of Next Instruction
		3	0	Stack Pointer	1	Irrelevant Data (NOTE 1)
		4	0	New Index Register	1	Irrelevant Data (NOTE 1)
TXS	4	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Op Code of Next Instruction
		3	0	Index Register	1	Irrelevant Data
		4	0	New Stack Pointer	1	Irrelevant Data
RTS	5	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Irrelevant Data (NOTE 2)
		3	0	Stack Pointer	1	Irrelevant Data (NOTE 1)
		4	1	Stack Pointer + 1	1	Address of Next Instruction (High Order Byte)
		5	1	Stack Pointer + 2	1	Address of Next Instruction (Low Order Byte)
WAI	9	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Op Code of Next Instruction
		3	1	Stack Pointer	0	Return Address (Low Order Byte)
		4	1	Stack Pointer – 1	0	Return Address (High Order Byte)
		5	1	Stack Pointer – 2	0	Index Register (Low Order Byte)
		6	1	Stack Pointer – 3	0	Index Register (High Order Byte)
		7	1	Stack Pointer – 4	0	Contents of Accumulator A
		8	1	Stack Pointer – 5	0	Contents of Accumulator B
		9	1	Stack Pointer – 6 (NOTE 3)	1	Contents of Cond. Code Register
RTI	10	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Irrelevant Data (NOTE 2)
		3	0	Stack Pointer	1	Irrelevant Data (NOTE 1)
		4	1	Stack Pointer + 1	1	Contents of Cond. Code Register from Stack
		5	1	Stack Pointer + 2	1	Contents of Accumulator B from Stack
		6	1	Stack Pointer + 3	1	Contents of Accumulator A from Stack
		7	1	Stack Pointer + 4	1	Index Register from Stack (High Order Byte)
		8	1	Stack Pointer + 5	1	Index Register from Stack (Low Order Byte)
		9	1	Stack Pointer + 6	1	Next Instruction Address from Stack (High Order Byte)
		10	1	Stack Pointer + 7	1	Next Instruction Address from Stack (Low Order Byte)
SWI	12	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Irrelevant Data (NOTE 1)
		3	1	Stack Pointer	0	Return Address (Low Order Byte)
		4	1	Stack Pointer – 1	0	Return Address (High Order Byte)
		5	1	Stack Pointer – 2	0	Index Register (Low Order Byte)
		6	1	Stack Pointer – 3	0	Index Register (High Order Byte)
		7	1	Stack Pointer – 4	0	Contents of Accumulator A
		8	1	Stack Pointer – 5	0	Contents of Accumulator B
		9	1	Stack Pointer – 6	0	Contents of Cond. Code Register
		10	0	Stack Pointer – 7	1	Irrelevant Data (NOTE 1)
		11	1	Vector Address FFFA (Hex)	1	Address of Subroutine (High Order Byte)
		12	1	Vector Address FFFB (Hex)	1	Address of Subroutine (Low Order Byte)

NOTE 1. If device which is addressed during this cycle uses VMA, then the Data Bus will go to the high impedance three-state condition. Depending on bus capacitance, data from the previous cycle may be retained on the Data Bus.
NOTE 2. Data is ignored by the MPU
NOTE 3. While the MPU is waiting for the interrupt, Bus Available will go "High" indicating the following states of the control lines: VMA is "Low"; Address Bus, R/$\overline{W}$, and Data Bus are all in the high impedance state.

Table 9 Immediate Mode Cycle by Cycle Operation

Address Mode and Instructions	Cycle	Cycle #	VMA Line	Address Bus	R/$\overline{W}$ Line	Data Bus
ADC EOR ADD LDA AND ORA BIT SBC CMP SUB	2	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Operand Data
CPX LDS LDX	3	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Operand Data (High Order Byte)
		3	1	Op Code Address + 2	1	Operand Data (Low Order Byte)

- **Direct and Extended Addressing Modes**

In the Direct and Extended modes of addressing, the operand field of the source statement is the address of the value that is to be operated on. The Direct and Extended modes differ only in the range of memory locations to which they can direct the MPU. Direct addressing generates a single 8-bit operand and, hence, can address only memory locations 0 ~ 255; a two byte operand is generated for Extended addressing, enabling the MPU to reach the remaining memory locations, 256 ~ 65535. An example of Direct addressing and its effect on program flow is illustrated in Fig. 34.

Table 10 shows the cycle-by-cycle operations of this mode.

The MPU, after encountering the opcode for the instrution LDAA (Direct) at memory location 5004 (Program Counter = 5004), looks in the next location, 5005, for the address of the operand. It then sets the program counter equal to the value found there (100 in the example) and fetches the operand, in this case a value to be loaded into accumulator A, from that location. For instructions requiring a two-byte operand such as LDX (Load the Index Register), the operand bytes would be retrieved from locations 100 and 101.

Extended addressing, Fig. 35, is similar except that a two-byte address is obtained from locations 5007 and 5008 after the LDAB (Extended) opcode shows up in location 5006. Extended addressing can be thought of as the "standard" addressing mode, that is, it is a method of reaching anyplace in memory. Direct addressing, since only one address byte is required, provides a faster method of processing data and generates fewer bytes of control code. In most applications, the direct addressing range, memory locations 0 ~ 255, are reserved for RAM. They are used for data buffering and temporary storage of system variables, the area in which faster addressing is of most value, Cycle-by-cycle operation is shown in Table 11 for Extended Addressing.

Table 10 Direct Mode Cycle by Cycle Operation

Address Mode and Instructions	Cycle	Cycle #	VMA Line	Address Bus	R/$\overline{W}$ Line	Data Bus
ADC EOR ADD LDA AND ORA BIT SBC CMP SUB	3	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Address of Operand
		3	1	Address of Operand	1	Operand Data
CPX LDS LDX	4	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Address of Operand
		3	1	Address of Operand	1	Operand Data (High Order Byte)
		4	1	Operand Address + 1	1	Operand Data (Low Order Byte)
STA	4	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Destination Address
		3	0	Destination Address	1	Irrelevant Data (NOTE 1)
		4	1	Destination Address	0	Data from Accumulator
STS STX	5	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Address of Operand
		3	0	Address of Operand	1	Irrelevant Data (NOTE 1)
		4	1	Address of Operand	0	Register Data (High Order Byte)
		5	1	Address of Operand + 1	0	Register Data (Low Order Byte)

NOTE 1. If device which is address during this cycle uses VMA, then the Data Bus will go to the high impedance three-state condition. Depending on bus capacitance, data from the previous cycle may be retained on the Data Bus.

Table 11 Extended Mode Cycle by Cycle

Address Mode and Instructions	Cycle	Cycle #	VMA Line	Address Bus	R/$\overline{W}$ Line	Data Bus
STS STX	6	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Address of Operand (High Order Byte)
		3	1	Op Code Address + 2	1	Address of Operand (Low Order Byte)
		4	0	Address of Operand	1	Irrelevant Data (NOTE 1)
		5	1	Address of Operand	0	Operand Data (High Order Byte)
		6	1	Address of Operand + 1	0	Operand Data (Low Order Byte)
JSR	9	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Address of Subroutine (High Order Byte)
		3	1	Op Code Address + 2	1	Address of Subroutine (Low Order Byte)
		4	1	Subroutine Starting Address	1	Op Code of Next Instruction
		5	1	Stack Pointer	0	Return Address (Low Order Byte)
		6	1	Stack Pointer − 1	0	Return Address (High Order Byte)
		7	0	Stack Pointer − 2	1	Irrelevant Data (NOTE 1)
		8	0	Op Code Address + 2	1	Irrelevant Data (NOTE 1)
		9	1	Op Code Address + 2	1	Address of Subroutine (Low Order Byte)
JMP	3	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Jump Address (High Order Byte)
		3	1	Op Code Address + 2	1	Jump Address (Low Order Byte)
ADC EOR ADD LDA AND ORA BIT SBC CMP SUB	4	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Address of Operand (High Order Byte)
		3	1	Op Code Address + 2	1	Address of Operand (Low Order Byte)
		4	1	Address of Operand	1	Operand Data
CPX LDS LDX	5	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Address of Operand (High Order Byte)
		3	1	Op Code Address + 2	1	Address of Operand (Low Order Byte)
		4	1	Address of Operand	1	Operand Data (High Order Byte)
		5	1	Address of Operand + 1	1	Operand Data (Low Order Byte)
STA A STA B	5	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Destination Address (High Order Byte)
		3	1	Op Code Address + 2	1	Destination Address (Low Order Byte)
		4	0	Operand Destination Address	1	Irrelevant Data (NOTE 1)
		5	1	Operand Destination Address	0	Data from Accumulator
ASL LSR ASR NEG CLR ROL COM ROR DEC TST INC	6	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Address of Operand (High Order Byte)
		3	1	Op Code Address + 2	1	Address of Operand (Low Order Byte)
		4	1	Address of Operand	1	Current Operand Data
		5	0	Address of Operand	1	Irrelevant Data (NOTE 1)
		6	1/0 (NOTE 2)	Address of Operand	0	New Operand Data (NOTE 2)

NOTE 1. If device which is addressed during this cycle uses VMA, then the Data Bus will go to the high impedance three-state condition. Depending on bus capacitance, data from the previous cycle may be retained on the Data Bus.
NOTE 2. For TST, VMA = 0 and Operand data does not change.

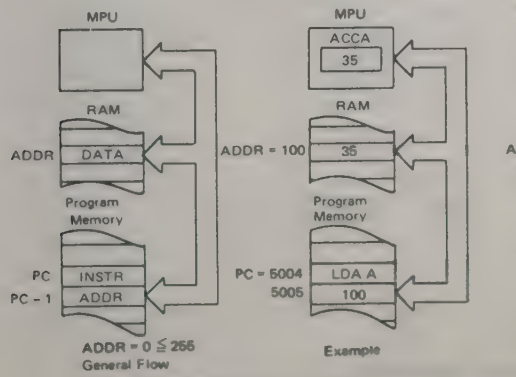


Figure 34 Direct Addressing Mode

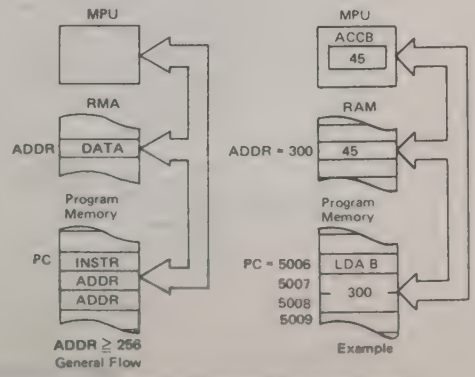


Figure 35 Extended Addressing Mode

- **Relative Address Mode**

In both the Direct and Extended modes, the address obtained by the MPU is an absolute numerical address. The Relative addressing mode, implemented for the MPU's branch instructions, specifies a memory location relative to the Program Counter's current location. Branch instructions generate two bytes of machine code, one for the instruction opcode and one for the "relative" address (see Fig. 36). Since it is desirable to be able to branch in either direction, the 8-bit address byte is interpreted as a signed 7-bit value; the 8th bit of the operand is treated as a sign bit, "0" = plus and "1" = minus. The remaining seven bits represent the numerical value. This result in a relative addressing range of ±127 with respect to the location of the branch instruction itself. However, the branch range is computed with respect to the next instruction that would be executed if the branch conditions are not satisfied. Since two byte are generated, the next instruction is located at PC+2. If, D is defined as the address of the branch destination, the range is then;

$$(PC+2) - 128 \leqq D \leqq (PC+2) + 127$$

or

$$PC - 126 \leqq D \leqq PC + 129$$

that is, the destination of the branch instruction must be within -126 to +129 memory locations of the branch instruction itself. For transferring control beyond this range, the unconditional jump (JMP), jump to subroutine (JSR), and return from subroutine (RTS) are used.

In Fig. 36, when the MPU encounters the opcode for BEQ (Branch if result of last instruction was zero), it tests the Zero bit in the Condition Code Register. If that bit is "0", indicating a non-zero result, the MPU continues execution with the next instruction (in location 5010 in Fig. 36). If the previous result was zero, the branch condition is satisfied and the MPU adds the offset, 15 in this case, to PC+2 and branches to location 5025 for the next instruction.

The branch instructions allow the programmer to efficiently direct the MPU to one point or another in the control program depending on the outcome of test results. Since the control program is normally in read-only memory and cannot be changed, the relative address used in execution of branch instructions is a constant numerical value. Cycle-by-cycle operation is shown in Table 12 for relative addressing.

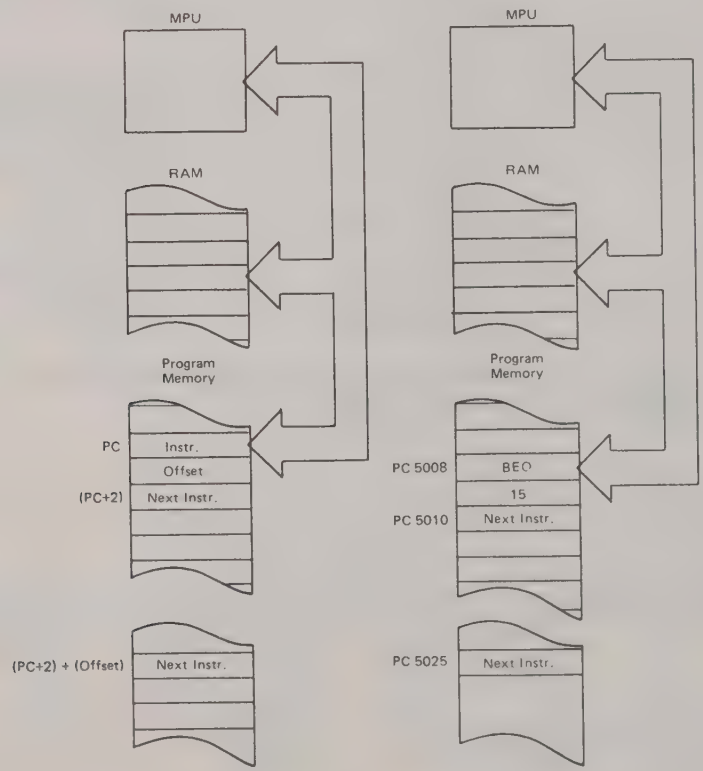


Figure 36 Relative Addressing Mode

Table 12 Relative Mode Cycle-by-Cycle Operation

Address Mode and Instructions	Cycle	Cycle #	VMA Line	Address Bus	R/$\overline{W}$ Line	Data Bus
BCC BHI BNE BCS BLE BPL BEQ BLS BRA BGE BLT BVC BGT BMI BVS	4	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Branch Offset
		3	0	Op Code Address + 2	1	Irrelevant Data (NOTE 1)
		4	0	Branch Address	1	Irrelevant Data (NOTE 1)
BSR	8	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Branch Offset
		3	0	Return Address of Main Program	1	Irrelevant Data (NOTE 1)
		4	1	Stack Pointer	0	Return Address (Low Order Byte)
		5	1	Stack Pointer − 1	0	Return Address (High Order Byte)
		6	0	Stack Pointer − 2	1	Irrelevant Data (NOTE 1)
		7	0	Return Address of Main Program	1	Irrelevant Data (NOTE 1)
		8	0	Subroutine Address	1	Irrelevant Data (NOTE 1)

NOTE 1. If device which is addressed during this cycle uses VMA, then the Data Bus will go to the high impedance three-state condition. Depending on bus capacitance, data from the previous cycle may be retained on the Data Bus.

- **Indexed Addressing Mode**

With Indexed addressing the numerical address is variable and depend on the current contents of the Index Register. A source statement such as

Operator	Operand	Comment
STAA	X	PUT A IN INDEXED LOCATION

causes the MPU to store the contents of accumulator A in the memory location specified by the contents of the Index Register (recall that the label X is reserved to designate the Index Register). Since there are instructions for manipulating X during program execution (LDX, INX, DEX, etc.), the Indexed addressing mode provides a dynamic "on the fly" way to modify program activity.

The operand field can also contain a numerical value that will be automatically added to X during execution. This format is illustrated in Fig. 37.

When the MPU encounters the LDAB (Indexed) opcode in location 5006, it looks in the next memory location for the value to be added to X (5 in the example) and calculates the required address by adding 5 to the present Index Register value of 400. In the operand format, the offset may be represented by a label or a numerical value in the range 0 ~ 255 as in the example. In the earlier example, STAA X, the operand is equivalent to 0, X , that is, the "0" may be omitted when the desired address is equal to X. Table 13 shows the cycle-by-cycle operation for the Indexed Mode of Addressing.

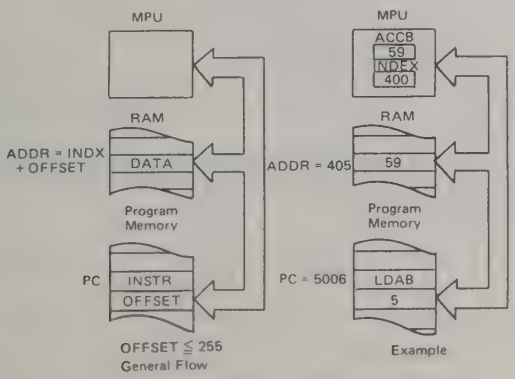


Figure 37 Indexed Addressing Mode

Table 13 Indexed Mode Cycle by Cycle

Address Mode and Instructions	Cycle	Cycle #	VMA Line	Address Bus	R/W Line	Data Bus
JMP	4	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Offset
		3	0	Index Register	1	Irrelevant Data (NOTE 1)
		4	0	Index Register Plus Offset (w/o Carry)	1	Irrelevant Data (NOTE 1)
ADC EOR ADD LDA AND ORA BIT SBC CMP SUB	5	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Offset
		3	0	Index Register	1	Irrelevant Data (NOTE 1)
		4	0	Index Register Plus Offset (w/o Carry)	1	Irrelevant Data (NOTE 1)
		5	1	Index Register Plus Offset	1	Operand Data
CPX LDS LDX	6	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Offset
		3	0	Index Register	1	Irrelevant Data (NOTE 1)
		4	0	Index Register Plus Offset (w/o Carry)	1	Irrelevant Data (NOTE 1)
		5	1	Index Register Plus Offset	1	Operand Data (High Order Byte)
		6	1	Index Register Plus Offset + 1	1	Operand Data (Low Order Byte)
STA	6	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Offset
		3	0	Index Register	1	Irrelevant Data (NOTE 1)
		4	0	Index Register Plus Offset (w/o Carry)	1	Irrelevant Data (NOTE 1)
		5	0	Index Register Plus Offset	1	Irrelevant Data (NOTE 1)
		6	1	Index Register Plus Offset	0	Operand Data
ASL LSR ASR NEG CLR ROL COM ROR DEC TST INC	7	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Offset
		3	0	Index Register	1	Irrelevant Data (NOTE 1)
		4	0	Index Register Plus Offset (w/o Carry)	1	Irrelevant Data (NOTE 1)
		5	1	Index Register Plus Offset	1	Current Operand Data
		6	0	Index Register Plus Offset	1	Irrelevant Data (NOTE 1)
		7	1/0 (NOTE 2)	Index Register Plus Offset	0	New Operand Data (NOTE 2)
STS STX	7	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Offset
		3	0	Index Register	1	Irrelevant Data (NOTE 1)
		4	0	Index Register Plus Offset (w/o Carry)	1	Irrelevant Data (NOTE 1)
		5	0	Index Register Plus Offset	1	Irrelevant Data (NOTE 1)
		6	1	Index Register Plus Offset	0	Operand Data (High Order Byte)
		7	1	Index Register Plus Offset + 1	0	Operand Data (Low Oder Byte)
JSR	8	1	1	Op Code Address	1	Op Code
		2	1	Op Code Address + 1	1	Offset
		3	0	Index Register	1	Irrelevant Data (NOTE 1)
		4	1	Stack Pointer	0	Return Address (Low Order Byte)
		5	1	Stack Pointer − 1	0	Return Address (High Order Byte)
		6	0	Stack Pointer − 2	1	Irrelevant Data (NOTE 1)
		7	0	Index Register	1	Irrelevant Data (NOTE 1)
		8	0	Index Register Plus Offset (w/o Carry)	1	Irrelevant Data (NOTE 1)

NOTE 1. If Device which is addressed during this cycle uses VMA, then the Data Bus will go to the high impedance three-state condition. Depending on bus capacitance, data from the previous cycle may be retained on the Data Bus.

NOTE 2. For TST, VMA = 0 and Operand data does not change.

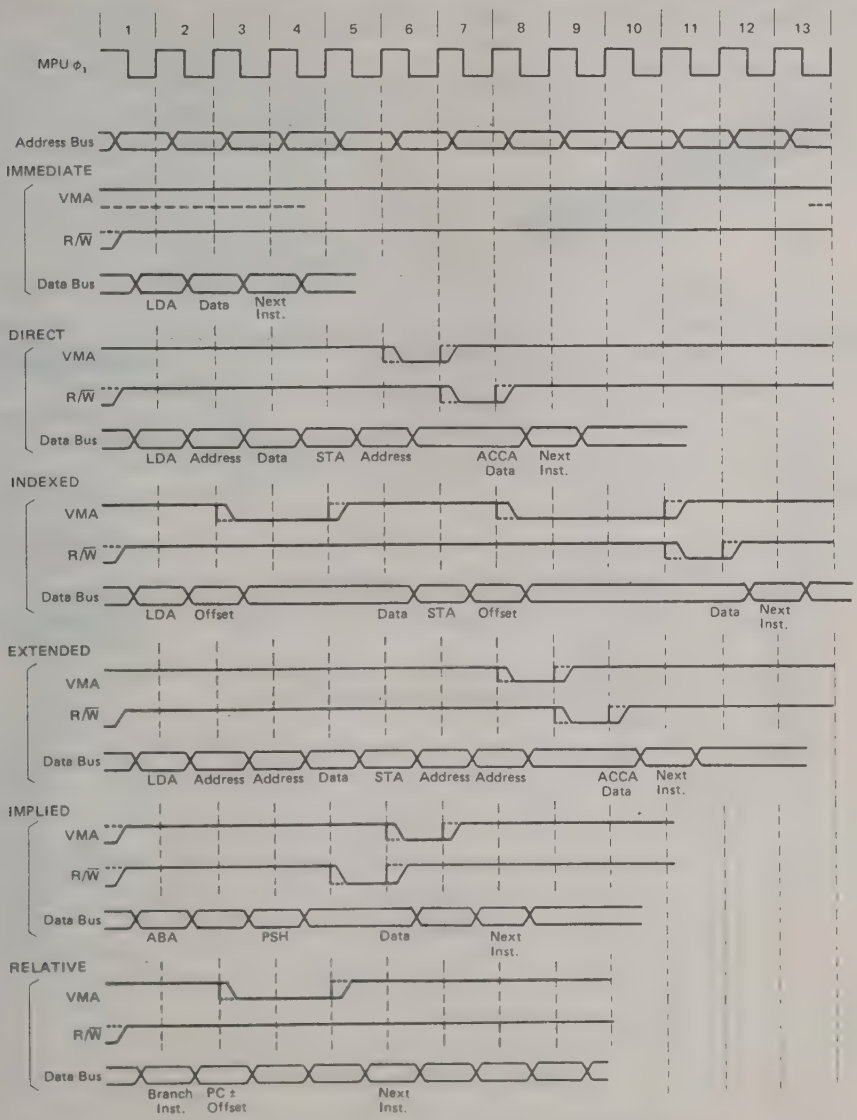


Figure 38 Example of Excution Timing in Each Addressing Mode

■ NOTE FOR THE RELATION BETWEEN WAI INSTRUCTION AND HALT OPERATION OF HD6800

When $\overline{\text{HALT}}$ input signal is asserted to "Low" level, the MPU will be halted after the execution of the current instruction except WAI instruction.

The "Halt" signal is not accepted after the fetch cycle of the WAI instruction (See ① in Fig. 39). In the case of the "WAI" instruction, the MPU enters the "WAIT" cycle after stacking the internal registers and outputs the "High" level on the BA line.

When an interrupt request signal is input to the MPU, the MPU accepts the interrupt regardless the "Halt" signal and releases the "WAIT" state and outputs the interrupt's vector address. If the "Halt" signal is "Low" level, the MPU halts after the fetch of new PC contents. The sequense is shown below.

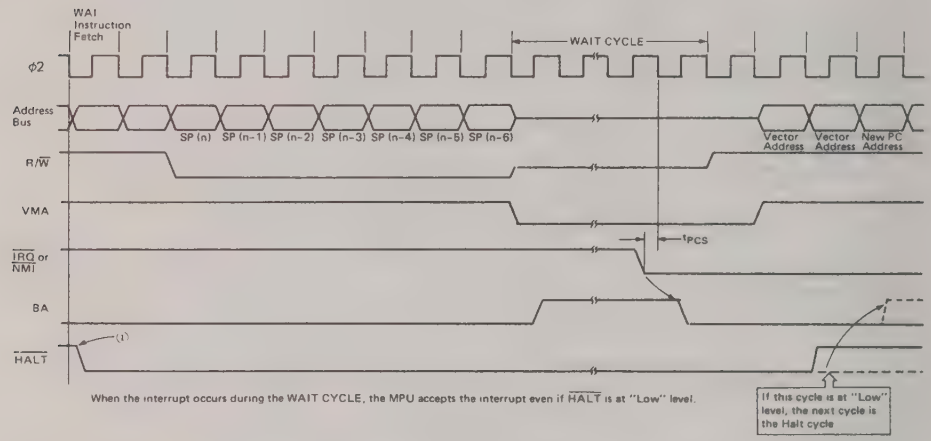

WAI Instruction Fetch
WAIT CYCLE
φ2
Address Bus
SP (n) SP (n-1) SP (n-2) SP (n-3) SP (n-4) SP (n-5) SP (n-6)
Vector Address
Vector Address
New PC Address
R/W
VMA
tPCS
IRQ or NMI
BA
HALT
①
When the interrupt occurs during the WAIT CYCLE, the MPU accepts the interrupt even if HALT is at "Low" level.
If this cycle is at "Low" level, the next cycle is the Halt cycle

Figure 39 HD6800 WAIT CYCLE & $\overline{\text{HALT}}$ Request

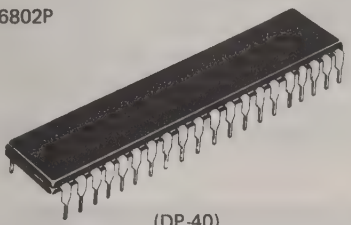
HD6802

MPU (Microprocessor with Clock and RAM)

The HD6802 is a monolithic 8-bit microprocessor that contains all the registers and accumulators of the present HD6800 plus an internal clock oscillator and driver on the same chip. In addition, the HD6802 has 128 bytes of RAM on the chip located at hex addresses 0000 to 007F. The first 32 bytes of RAM, at hex addresses 0000 to 001F, may be retained in a low power mode by utilizing V_{CC} standby, thus facilitating memory retention during a power-down situation.

The HD6802 is completely software compatible with the HD6800 as well as the entire HMCS6800 family of parts. Hence, the HD6802 is expandable to 65k words.

HD6802P

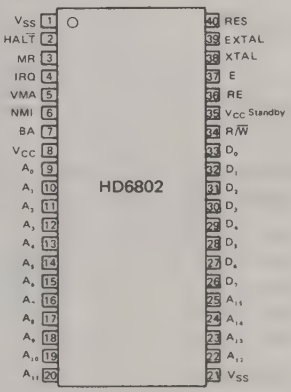


(DP-40)

■ FEATURES

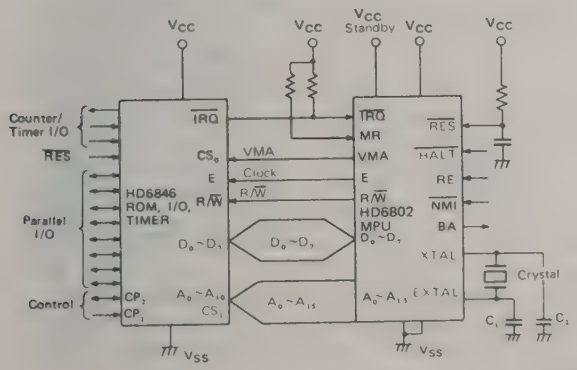
- On-Chip Clock Circuit
- 128 × 8 Bit On-Chip RAM
- 32 Bytes of RAM are Retainable
- Software-Compatible with the HD6800
- Expandable to 65k words
- Standard TTL-Compatible Inputs and Outputs
- 8 Bit Word Size
- 16 Bit Memory Addressing
- Interrupt Capability
- Compatible with MC6802

■ PIN ARRANGEMENT



(Top View)

■ BLOCK DIAGRAM



■ ABSOLUTE MAXIMUM RATINGS

Item	Symbol	Value	Unit
Supply Voltage	V_{CC} * V_{CC} Standby*	-0.3 ~ +7.0	V
Input Voltage	V_{in} *	-0.3 ~ +7.0	V
Operating Temperature	T_{opr}	-20 ~ +75	°C
Storage Temperature	T_{stg}	-55 ~ +150	°C

* With respect to V_{SS} (SYSTEM GND)

(NOTE) Permanent LSI damage may occur if maximum ratings are exceeded. Normal operation should be under recommended operating conditions. If these conditions are exceeded, it could affect reliability of LSI.

■ RECOMMENDED OPERATING CONDITIONS

Item	Symbol		min	typ	max	Unit
Supply Voltage	V_{CC} * V_{CC} Standby*		4.75	5.0	5.25	V
Input Voltage	V_{IL} *		-0.3	–	0.8	V
	V_{IH} *	Except $\overline{RES}$	2.0	–	V_{CC}	V
		$\overline{RES}$	4.25	–	V_{CC}	V
Operation Temperature	T_{opr}		-20	25	75	°C

* With respect to V_{SS} (SYSTEM GND)

■ ELECTRICAL CHARACTERISTICS

● **DC CHARACTERISTICS (V_{CC}=5.0V±5%, V_{CC} Standby=5.0V±5%, V_{SS}=0V, Ta=-20~+75°C, unless otherwise noted.)**

Item		Symbol	Test Condition	min	typ**	max	Unit
Input "High" Voltage	Except $\overline{RES}$	V_{IH}		2.0	–	V_{CC}	V
	$\overline{RES}$			4.25	–	V_{CC}	
Input "Low" Voltage	Except $\overline{RES}$	*** V_{IL}		-0.3	–	0.8	V
	$\overline{RES}$			-0.3	–	0.8	
Output "High" Voltage	D_0~D_7, E	V_{OH}	I_{OH} = -205μA	2.4	–	–	V
	A_0~A_{15}, R/$\overline{W}$, VMA		I_{OH} = -145μA	2.4	–	–	
	BA		I_{OH} = -100μA	2.4	–	–	
Output "Low" Voltage		V_{OL}	I_{OL} = 1.6mA	–	–	0.4	V
Three State (Off State) Input Current	D_0~D_7	I_{TSI}	V_{in} = 0.4~2.4V	-10	–	10	μA
Input Leakage Current	Except D_0~D_7 ****	I_{in}	V_{in} = 0~5.25V	-2.5	–	2.5	μA
Power Dissipation		P_D*		–	0.6	1.2	W
Input Capacitance	D_0~D_7	C_{in}	V_{in}=0V, T_a=25°C, f=1.0MHz	–	10	12.5	pF
	Except D_0~D_7			–	6.5	10	
Output Capacitance	A_0~A_{15}, R/$\overline{W}$, BA, VMA, E	C_{out}	V_{in}=0V, T_a=25°C, f=1.0MHz	–	–	12	pF

* In power-down mode, maximum power dissipation is less than 42mW.
** T_a=25° C, V_{CC} =5V
*** As $\overline{RES}$ input has histeresis character, applied voltage up to 2.4V is regarded as "Low" level when it goes up from 0V.
**** Does not include EXTAL and XTAL, which are crystal inputs.

• AC CHARACTERISTICS (V_{CC}=5.0V±5%, V_{CC} Standby=5.0V±5%, V_{SS}=0V, Ta=-20~+75°C, unless otherwise noted.)

1. CLOCK TIMING CHARACTERISTICS

Item		Symbol	Test Condition	min	typ	max	Unit
Frequency of Operation	Input Clock ÷ 4	f		0.1	–	1.0	MHz
	Crystal Frequency	f_{XTAL}		1.0	–	4.0	
Cycle Time		t_{cyc}	Fig. 2, Fig. 3	1.0	–	10	μs
Clock Pulse Width	"High" Level	$PW_{\phi H}$	at 2.4V (Fig. 2, Fig. 3)	450	–	4500	ns
	"Low" Level	$PW_{\phi L}$	at 0.8V (Fig. 2, Fig. 3)				
Clock Fall Time		t_{ϕ}	0.8V ~ 2.4V(Fig.2,Fig.3)	–	–	25	ns

2. READ/WRITE TIMING

Item	Symbol	Test Condition	min	typ*	max	Unit
Address Delay	t_{AD}	Fig. 2, Fig. 3, Fig. 6	–	–	270	ns
Peripheral Read Access Time	t_{acc}	Fig. 2	530	–	–	ns
Data Setup Time (Read)	t_{DSR}	Fig. 2	100	–	–	ns
Input Data Hold Time	t_{H}	Fig. 2	10	–	–	ns
Output Data Hold Time	t_{H}	Fig. 3	20	–	–	ns
Address Hold Time (Address, R/$\overline{W}$, VMA)	t_{AH}	Fig. 2, Fig. 3	10	–	–	ns
Data Delay Time (Write)	t_{DDW}	Fig. 3	–	–	225	ns
Bus Available Delay	t_{BA}	Fig. 4, Fig. 5, Fig. 7, Fig. 8	–	–	250	ns
Processor Controls Processor Control Setup Time	t_{PCS}	Fig. 4~Fig. 7, Fig. 12	200	–	–	ns
Processor Control Rise and Fall Time (Measured at 0.8V and 2.0V)	t_{PCr} t_{PCf}	Fig. 4~Fig. 7, Fig. 12, Fig. 13, Fig. 16	–	–	100	ns

*Ta = 25°C, V_{CC} = 5V

3. POWER DOWN SEQUENCE TIMING, POWER UP RESET TIMING AND MEMORY READY TIMING

Item	Symbol	Test Condition	min	typ	max	Unit
RAM Enable Reset TIme (1)	t_{RE1}	Fig. 13	150	–	–	ns
RAM Enable Reset Time (2)	t_{RE2}	Fig. 13	E-3 cycles	–	–	
Reset Release Time	t_{LRES}	Fig. 12	20*	–	–	ms
RAM Enable Reset Time (3)	t_{RE3}	Fig. 12	0	–	–	ns
Memory Ready Setup Time	t_{SMR}	Fig. 16	300	–	–	ns
Memory Ready Hold Time	t_{HMR}	Fig. 16	0	–	200	ns

*t_{RES} = 20 msec min. for S type, 50 msec min. for R type.

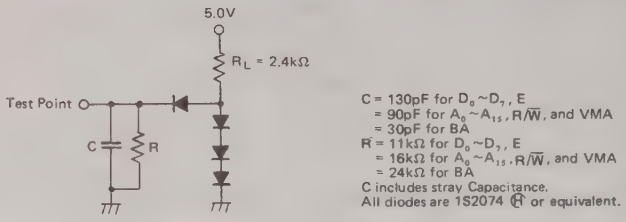

5.0V
R_L = 2.4kΩ
Test Point
C
R
C = 130pF for D_0~D_7, E
= 90pF for A_0~A_{15}, R/$\overline{W}$, and VMA
= 30pF for BA
R = 11kΩ for D_0~D_7, E
= 16kΩ for A_0~A_{15}, R/$\overline{W}$, and VMA
= 24kΩ for BA
C includes stray Capacitance.
All diodes are 1S2074 (H) or equivalent.

Figure 1 Bus Timing Test Load

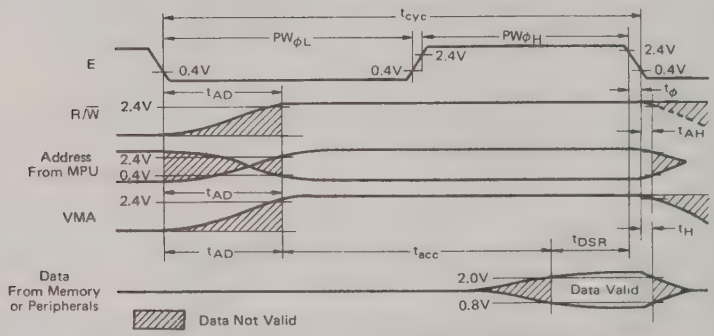


Figure 2 Read Data from Memory or Peripherals

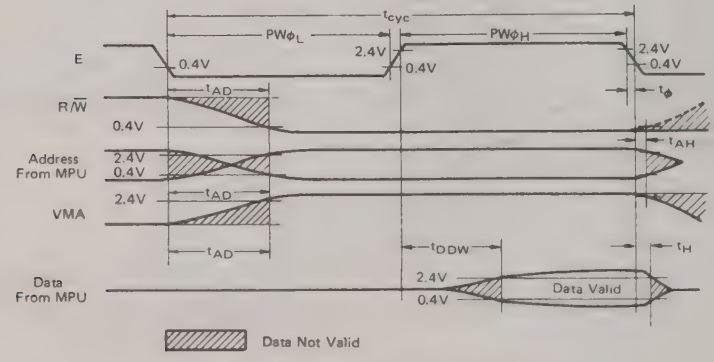


Figure 3 Write Data in Memory or Peripherals

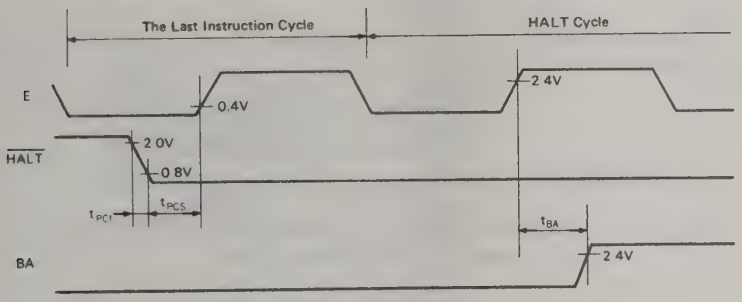


Figure 4 Timing of $\overline{\text{HALT}}$ and BA

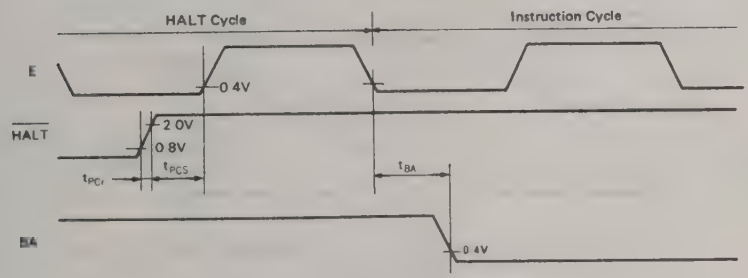


Figure 5 Timing of $\overline{\text{HALT}}$ and BA

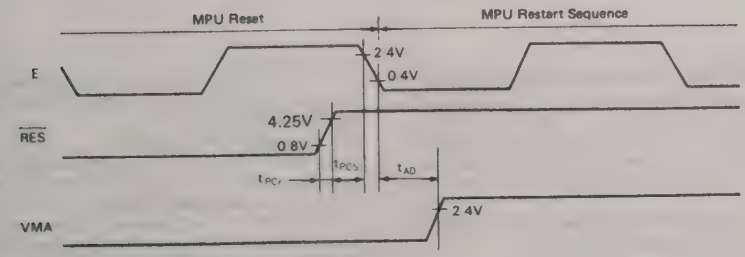


Figure 6 $\overline{\text{RES}}$ and MPU Restart Sequence

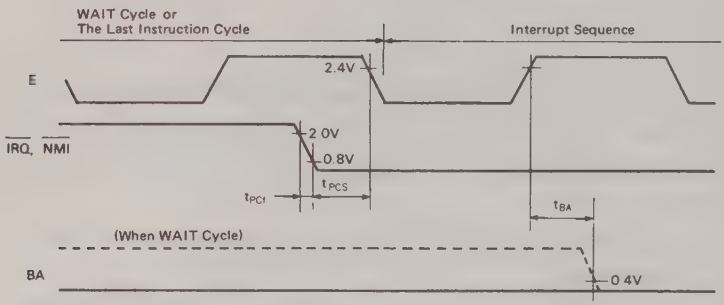


Figure 7 $\overline{IRQ}$ and $\overline{NMI}$ Interrupt Timing

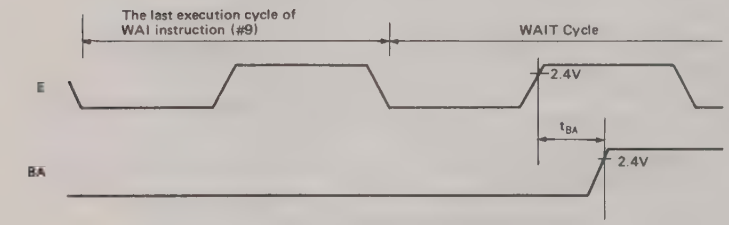


Figure 8 WAI Instruction and BA Timing

- **MPU REGISTERS**

A general block diagram of the HD6802 is shown in Fig. 9. As shown, the number and configuration of the registers are the same as for the HD6800. The 128 x 8 bit RAM has been added to the basic MPU. The first 32 bytes may be operated in a low power mode via a V_{CC} standby. These 32 bytes can be retained during power-up and power-down conditions via the RE signal.

The MPU has three 16-bit registers and three 8-bit registers available for use by the programmer (Fig. 10).

- **Program Counter (PC)**

The program counter is a two byte (16-bit) register that points to the current program address.

- **Stack Pointer (SP)**

The stack pointer is a two byte (16-bit) register that contains the address of the next available location in an external push-down/pop-up stack. This stack is normally a random access Read/Write memory that may have any location (address) that is convenient. In those applications that require storage of information in the stack when power is lost, the stack must be non-volatile.

- **Index Register (IX)**

The index register is a two byte register that is used to store data or a sixteen bit memory address for the Indexed mode of memory addressing.

- **Accumulators (ACCA, ACCB)**

The MPU contains two 8-bit accumulators that are used to hold operands and results from an arithmetic logic unit(ALU).

- **Condition Code Register (CCR)**

The condition code register indicates the results of an Arithmetic Logic Unit operation: Negative(N), Zero(Z), Overflow(V), Carry from bit7(C), and half carry from bit3(H). These bits of the Condition Code Register are used as testable conditions for the conditional branch instructions. Bit 4 is the interrupt mask bit(I). The used bits of the Condition Code Register (B6 and B7) are ones.

Fig. 11 shows the order of saving the microprocessor status within the stack.

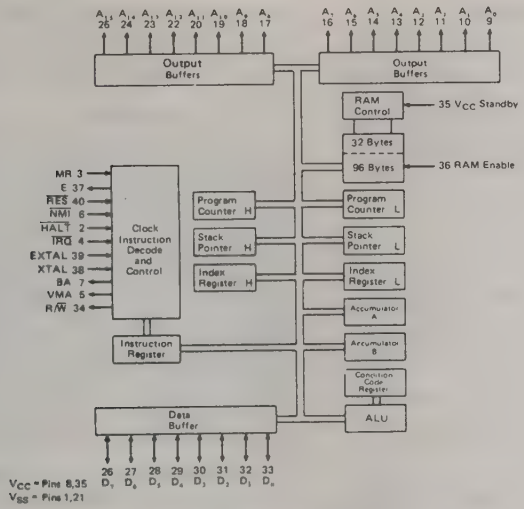


Figure 9 Expanded Block Diagram

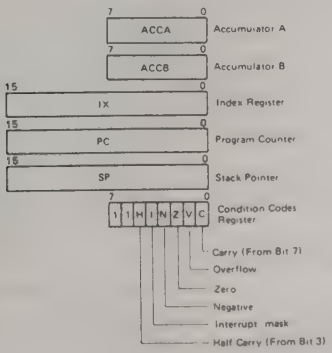


Figure 10 Programming Model of The Microprocessing Unit

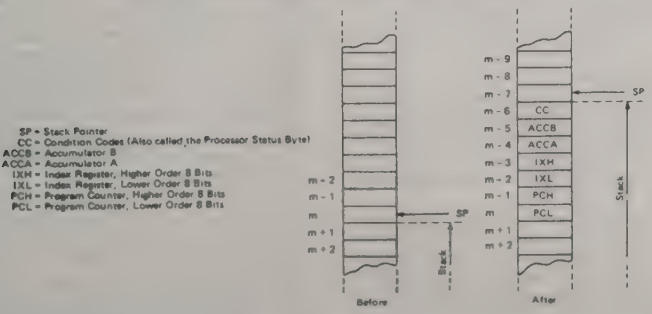

SP = Stack Pointer
CC = Condition Codes (Also called the Processor Status Byte)
ACCB = Accumulator B
ACCA = Accumulator A
IXH = Index Register, Higher Order 8 Bits
IXL = Index Register, Lower Order 8 Bits
PCH = Program Counter, Higher Order 8 Bits
PCL = Program Counter, Lower Order 8 Bits
Before
After
Stack
SP

Figure 11 Saving The Status of The Microprocessor in The Stack

■ HD6802 MPU SIGNAL DESCRIPTION

Proper operation of the MPU requires that certain control and timing signals be provided to accomplish specific functions and that other signal lines be monitored to determine the state of the processor. These control and timing signals for the HD6802 are similar to those of the HD6800 except that TSC, DBE, ϕ_1, ϕ_2 input, and two unused pins have been eliminated, and the following signal and timing lines have been added.

RAM Enable (RE)
Crystal Connections EXTAL and XTAL
Memory Ready(MR)
V_{CC} Standby
Enable ϕ_2 Output(E)

The following is a summary of the HD6802 MPU signals:

- **Address Bus ($A_0 \sim A_{15}$)**

Sixteen pins are used for the address bus. The outputs are capable of driving one standard TTL load and 90pF.

- **Data Bus ($D_0 \sim D_7$)**

Eight pins are used for the data bus. It is bidirectional, transferring data to and from the memory and peripheral devices. It also has three-state output buffers capable of driving one standard TTL load and 130pF.

Data Bus will be in the output mode when the internal RAM is accessed. This prohibits external data entering the MPU. It should be noted that the internal RAM is fully decoded from $0000 to $007F. External RAM at $0000 to $007F must be disabled when internal RAM is accessed.

- **$\overline{HALT}$**

When this input is in the "Low" state, all activity in the machine will be halted: This input is level sensitive.

In the halt mode, the machine will stop at the end of an instruction. Bus Available will be at a "High" state. Valid Memory Address will be at a "Low" state. The address bus will display the address of the next instruction.

To insure single instruction operation, transition of the $\overline{HALT}$ line must not occur during the last 250ns of E and the $\overline{HALT}$ line must go "High" for one Clock cycle.

$\overline{HALT}$ should be tied "High" if not used. This is good engineering design practice in general and necessary to insure proper operation of the part.

- **Read/Write (R/$\overline{W}$)**

This TTL compatible output signals the peripherals and memory devices whether the MPU is in a Read ("High") or Write ("Low") state. The normal standby state of this signal is Read ("High"). When the processor is halted, it will be in the logical one state ("High").

This output is capable of driving one standard TTL load and 90pF.

- **Valid Memory Address (VMA)**

This output indicates to peripheral devices that there is a valid address on the address bus. In normal operation, this signal should be utilized for enabling peripheral interfaces such as the PIA and ACIA. This signal is not three-state. One standard TTL load and 90pF may be directly driven by this active high signal.

- **Bus Available (BA)**

The Bus Available signal will normally be in the "Low" state. When activated, it will go to the "High" state indicating that the microprocessor has stopped and that the address bus is available (but not in a three-state condition). This will occur if the $\overline{HALT}$ line is in the "Low" state or the processor is in the wait state as a result of the execution of a WAI instruction. At such time, all three-state output drivers will go to their off state and other outputs to their normally inactive level.

The processor is removed from the wait state by the occurrence of a maskable (mask bit I=0) or nonmaskable interrupt. This output is capable of driving one standard TTL load and 30pF.

- **Interrupt Request ($\overline{IRQ}$)**

This level sensitive input requests that an interrupt sequence be generated within the machine. The processor will wait, until it completes the current instruction that is being executed before it recognizes the request. At that time, if the interrupt mask bit in the Condition Code Register is not set, the machine will begin an interrupt sequence. The index Register, Program Counter, Accumulators, and Condition Code Register are stored away on the stack. Next the MPU will respond to the interrupt request by setting the interrupt mask bit high so that no further interrupts may occur. At the end of the cycle, a 16-bit address will be loaded that points to a vectoring address which is located in memory locations FFF8 and FFF9. An address loaded at these locations causes the MPU to branch to an interrupt routine in memory.

The $\overline{HALT}$ line must be in the "High" state for interrupts to be serviced. Interrupts will be latched internally while $\overline{HALT}$ is "Low".

A 3kΩ external register to V_{CC} should be used for wire-OR and optimum control of interrupts.

- **Reset ($\overline{RES}$)**

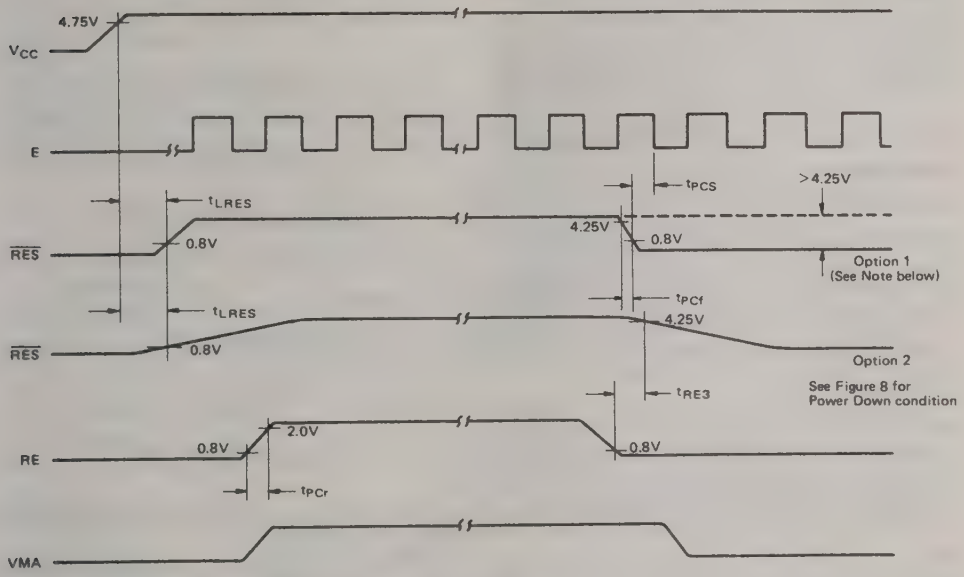
This input is used to reset and start the MPU from a power-down condition, resulting from a power failure or an initial start-up of the processor. When this line is "Low", the MPU is inactive and the information in the registers will be lost. If a "High" level is detected on the input, this will signal the MPU to begin the restart sequence. This will start execution of a routine to initialize the processor from its reset condition. All the higher order address lines will be forced "High". For the restart, the last two(FFFE, FFFF) locations in memory will be used to load the program that is addressed by the program counter. During the restart routine, the interrupt mask bit is set and must be reset before the MPU can be interrupted by $\overline{IRQ}$. Power-up and reset timing and power-down sequences are shown in Fig. 12 and Fig. 13 respectively.

- **Non-Maskable Interrupt ($\overline{NMI}$)**

A low-going edge on this input requests that a non-mask-interrupt sequence be generated within the processor. As with the $\overline{IRQ}$ signal, the processor will complete the current instruction that is being executed before it recognizes the $\overline{NMI}$ signal. The interrupt mask bit in the Condition Code Register has no effect on $\overline{NMI}$.

The Index Register, Program Counter, Accumulators, and Condition Code Register are stored away on the stack. At the end of the cycle, a 16-bit address will be loaded that points to a vectoring address which is located in memory locations FFFC and FFFD. An address loaded at these locations causes the MPU to branch to a non-maskable interrupt routine in memory. A 3kΩ external resistor to V_{CC} should be used for wire-OR and optimum control of interrupts.

Inputs $\overline{IRQ}$ and $\overline{NMI}$ are hardware interrupt lines that are sampled when E is "High" and will start the interrupt routine on a "Low" E following the completion of an instruction. $\overline{IRQ}$ and $\overline{NMI}$ should be tied "High" if not used. This is good engineering design practice in general and necessary to insure proper operation of the part. Fig. 14 is a flowchart describing the major decision paths and interrupt vectors of the microprocessor. Table 1 gives the memory map for interrupt vectors.



(NOTE) If option 1 is chosen, $\overline{RES}$ and RE pins can be tied together.

Figure 12 Power-up and Reset Timing

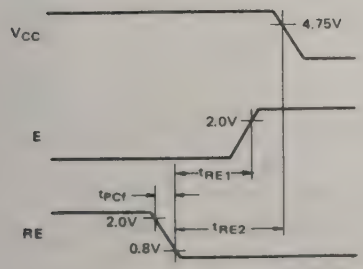


Figure 13 Power-down Sequence

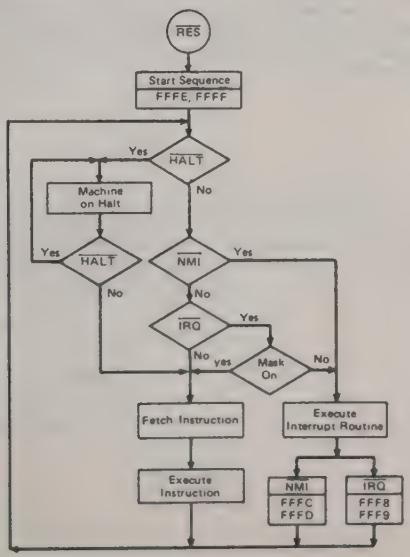


Figure 14 MPU Flow Chart

Table 1 Memory Map for Interrupt Vectors

Vector MS	Vector LS	Description	
FFFE	FFFF	Restart	($\overline{RES}$)
FFFC	FFFD	Non-Maskable Interrupt	($\overline{NMI}$)
FFFA	FFFB	Software Interrupt	(SWI)
FFF8	FFF9	Interrupt Request	($\overline{IRQ}$)

- **RAM Enable (RE)**

A TTL-compatible RAM enable input controls the on-chip RAM of the HD6802. When placed in the "High" state, the on-chip memory is enabled to respond to the MPU controls. In the "Low" state, RAM is disabled. This pin may also be utilized to disable reading and writing the on-chip RAM during a power-down situation. RAM enable must be "Low" three cycles before V_{CC} goes below 4.75V during power-down.

RE should be tied to the correct "High" or "Low" state if not used. This is good engineering design practice in general and necessary to insure proper operation of the part.

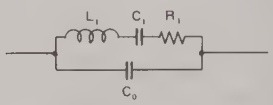
- **EXTAL and XTAL**

The HD6802 has an internal oscillator that may be crystal controlled. These connections are for a parallel resonant fundamental crystal (AT cut). A divide-by-four circuit has been added to the HD6802 so that a 4MHz crystal may be used in lieu of a 1MHz crystal for a more cost-effective system. Pin39 of the HD6802 may be driven externally by a TTL input signal if a separate clock is required. Pin38 is to be left open in this mode.

An RC network is not directly usable as a frequency source on pins 38 and 39. An RC network type TTL or CMOS oscillator will work well as long as the TTL or CMOS output drives the HD6802.

If an external clock is used, it may not be halted for more than 4.5μs. The HD6802 is a dynamic part except for the internal RAM, and requires the external clock to retain information.

Conditions for Crystal (4 MHz)
- AT Cut Parallel resonant
- C_0 = 7 pF max.
- R_1 = 80Ω max.



Crystal Equivalent Circuit

Recommended Oscillator (4MHz)

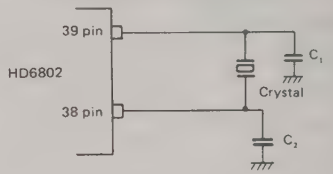


Figure 15 Crystal Oscillator

When using the crystal, see the note for Board Design of the Oscillation Circuit in HD6802.

- **Memory Ready (MR)**

MR is a TTL compatible input control signal which allows stretching of E. When MR is "High", E will be in normal operation. When MR is "Low", E may be stretched integral multiples of half periods, thus allowing interface to slow memories. Memory Ready timing is shown in Fig. 16.

MR should be tied "High" if not used. This is good engineering design practice in general and necessary to insure proper operation of the part. A maximum stretch is 4.5μs.

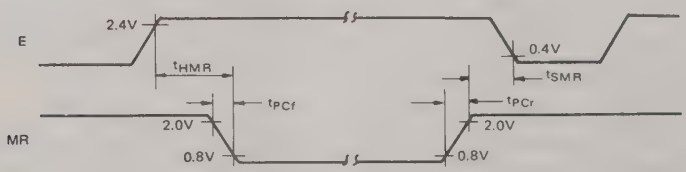


Figure 16 Memory Ready Control Function

- **Enable (E)**

This pin supplies the clock for the MPU and the rest of the system. This is a single phase, TTL compatible clock. This clock may be conditioned by a Memory Ready Signal. This is equivalent to ϕ_2 on the HD6800.

- **V_{CC} Standby**

This pin supplies the dc voltage to the first 32 bytes of RAM as well as the RAM Enable (RE) control logic. Thus retention of data in this portion of the RAM on a power up, power-down, or standby condition is guaranteed at the range of 4.0 V to 5.25 V.

Maximum current drain at 5.25V is 8mA.

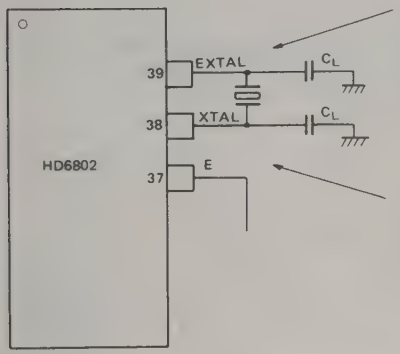
■ MPU INSTRUCTION SET

The HD6802 has a set of 72 different instructions. Included are binary and decimal arithmetic, logical, shift, rotate, load, store, conditional or unconditional branch, interrupt and stack manipulation instructions.

This instruction set is the same as that for the 6800MPU(HD6800 etc.) and is not explained again in this data sheet.

■ NOTE FOR BOARD DESIGN OF THE OSCILLATION CIRCUIT IN HD6802

In designing the board, the following notes should be taken when the crystal oscillator is used.

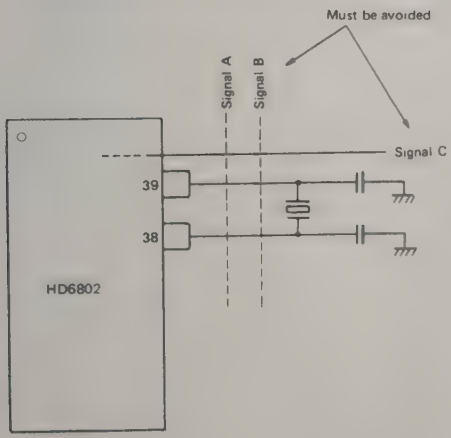


Crystal oscillator and load capacity C_L must be placed near the LSI as much as possible.

Normal oscillation may be disturbed when external noise is induced to pin 38 and 39.

Pin 38 signal line should be wired apart from pin 37 signal line as much as possible. Don't wire them in parallel, or normal oscillation may be disturbed when E signal is feedbacked to XTAL.

The following design must be avoided.



A signal line or a power source line must not cross or go near the oscillation circuit line as shown in the left figure to prevent the induction from these lines and perform the correct oscillation. The resistance among XTAL, EXTAL and other pins should be over 10MΩ.

Figure 17 Note for Board Design of the Oscillation Circuit

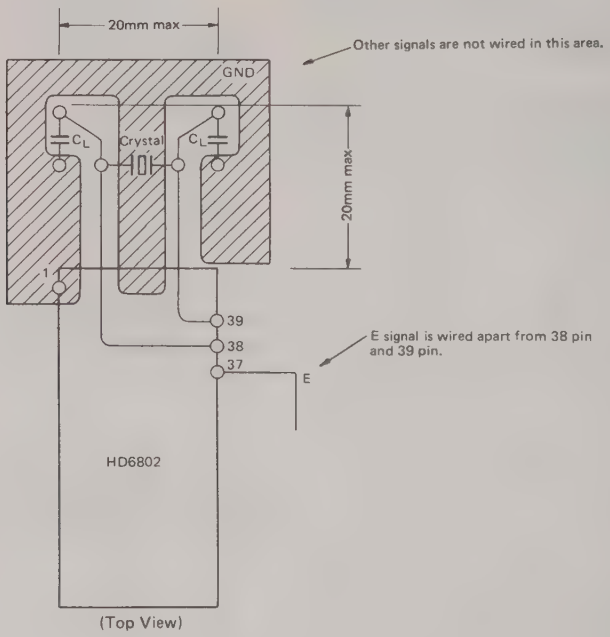


Figure 18 Example of Board Design Using the Crystal Oscillator

■ NOTE FOR THE RELATION BETWEEN WAI INSTRUCTION AND HALT OPERATION OF HD6802

When $\overline{HALT}$ input signal is asserted to "Low" level, the MPU will be halted after the execution of the current instruction except WAI instruction.

The "Halt" signal is not accepted after the fetch cycle of the WAI instruction (See ① in Fig. 19). In the case of the "WAI" instruction, the MPU enters the "WAIT" cycle after stacking the internal registers and outputs the "High" level on the BA line.

When an interrupt request signal is input to the MPU, the MPU accepts the interrupt regardless the "Halt" signal and releases the "WAIT" state and outputs the interrupt's vector address. If the "Halt" signal is "Low" level, the MPU halts after the fetch of new PC contents. The sequense is shown below.

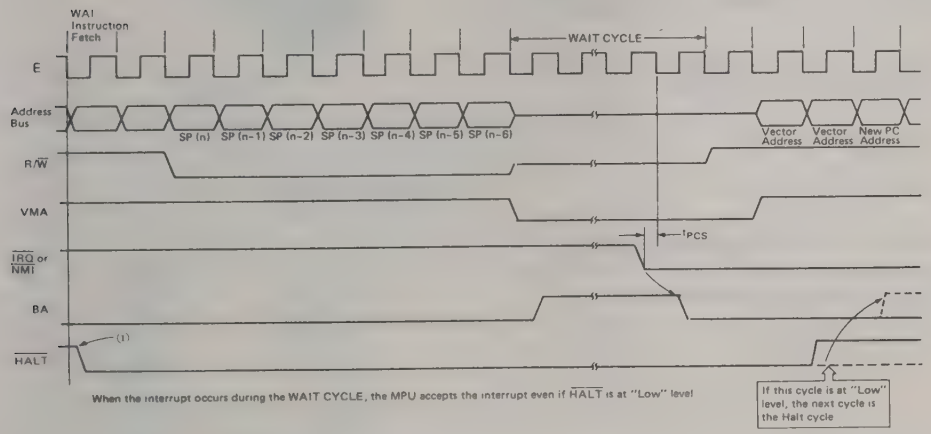

WAI Instruction Fetch
WAIT CYCLE
E
Address Bus
SP (n) SP (n-1) SP (n-2) SP (n-3) SP (n-4) SP (n-5) SP (n-6)
Vector Address
Vector Address
New PC Address
R/W
VMA
tPCS
IRQ or NMI
BA
HALT
①
When the interrupt occurs during the WAIT CYCLE, the MPU accepts the interrupt even if HALT is at "Low" level
If this cycle is at "Low" level, the next cycle is the Halt cycle

Figure 19 HD6802 WAIT CYCLE & $\overline{HALT}$ Request

HD6802W

MPU (Microprocessor with Clock and RAM)

HD6802W is the enhanced version of HD6802 which contains MPU, clock and 256 bytes RAM. Internal RAM has been extended from 128 to 256 bytes to increase the capacity of system read/write memory for handling temporary data and manipulating the stack.

The internal RAM is located at hex addresses 0000 to 00FF. The first 32 bytes of RAM, at hex addresses 0000 to 001F, may be retained in a low power mode by utilizing V_{CC} standby, thus facilitating memory retention during a power-down situation.

The HD6802W is completely software compatible with the HD6800 as well as the entire HMCS6800 family of parts. Hence, the HD6802W is expandable to 65k words.

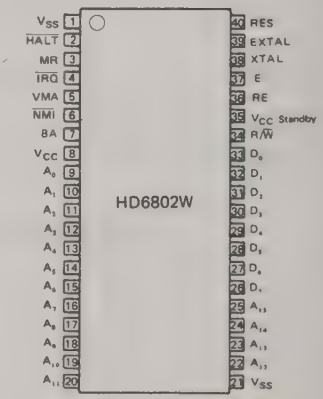
HD6802WP

(DP-40)

■ FEATURES

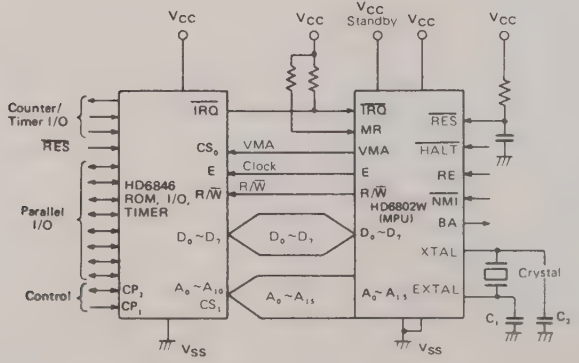
- On-Chip Clock Circuit
- 256 × 8 Bit On-Chip RAM
- 32 Bytes of RAM are Retainable
- Software-Compatible with the HD6800, HD6802
- Expandable to 65k words
- Standard TTL-Compatible Inputs and Outputs
- 8 Bit Word Size
- 16 Bit Memory Addressing
- Interrupt Capability

■ PIN ARRANGEMENT



(Top View)

■ BLOCK DIAGRAM



A expanded block diagram of the HD6802W is shown in Fig. 1. As shown, the number and configuration of the registers are the same as the HD6802 except that the internal RAM has been extended to 256 bytes.

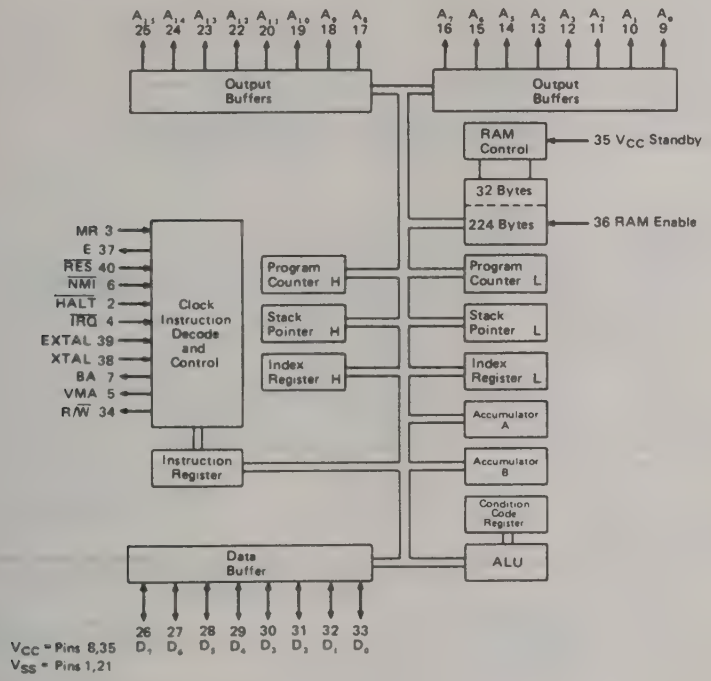


Figure 1 Expanded Block Diagram

Address Map of RAM is shown is Fig. 2.

The HD6802W has 256 bytes of RAM on the chip located at hex addresses 0000 to 00FF. The first 32 bytes of RAM, at hex addresses 0000 to 001F, may be retained in a low power mode by utilizing V_{CC} standby and setting RAM Enable Signal "Low" level, thus facilitating memory retention during a power-down situation.

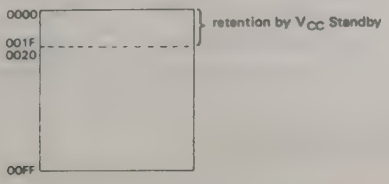


Figure 2 Address Map of HD6802W

■ ABSOLUTE MAXIMUM RATINGS

Item	Symbol	Value	Unit
Supply Voltage	V_{CC} * V_{CC} Standby*	-0.3 ~ +7.0	V
Input Voltage	V_{in}*	-0.3 ~ +7.0	V
Operating Temperature	T_{opr}	-20 ~ +75	°C
Storage Temperature	T_{stg}	-55 ~ +150	°C

* With respect to V_{SS} (SYSTEM GND)

(NOTE) Permanent LSI damage may occur if maximum ratings are exceeded. Normal operation should be under recommended operating conditions. If these conditions are exceeded, it could affect reliability of LSI.

■ RECOMMENDED OPERATING CONDITIONS

Item	Symbol		min	typ	max	Unit
Supply Voltage	V_{CC}*		4.75	5.0	5.25	V
	V_{CC} Standby*		4.0			
Input Voltage	V_{IL}*		-0.3	–	0.8	V
	V_{IH}*	Except $\overline{RES}$	2.0	–	V_{CC}	V
		$\overline{RES}$	V_{CC} -0.75	–	V_{CC}	
Operation Temperature	T_{opr}		-20	25	75	°C

* With respect to V_{SS} (SYSTEM GND)

■ ELECTRICAL CHARACTERISTICS

● DC CHARACTERISTICS (V_{CC}=5.0V±5%, V_{CC} Standby=5.0V±5%, V_{SS}=0V, Ta=-20~+75°C, unless otherwise noted.)

Item		Symbol	Test Condition	min	typ*	max	Unit
Input "High" Voltage	Except $\overline{RES}$	V_{IH}		2.0	–	V_{CC}	V
	$\overline{RES}$			V_{CC}-0.75	–	V_{CC}	
Input "Low" Voltage	Except $\overline{RES}$	V_{IL}**		-0.3	–	0.8	V
	$\overline{RES}$			-0.3	–	0.8	
Output "High" Voltage	D_0~D_7, E	V_{OH}	I_{OH} = -205μA	2.4	–	–	V
	A_0~A_{15}, R/$\overline{W}$, VMA		I_{OH} = -145μA	2.4	–	–	
	BA		I_{OH} = -100μA	2.4	–	–	
Output "Low" Voltage		V_{OL}	I_{OL} = 1.6mA	–	–	0.4	V
Three State (Off State) Input Current	D_0~D_7	I_{TSI}	V_{in} = 0.4~2.4V	-10	–	10	μA
Input Leakage Current	Except D_0~D_7	I_{in}***	V_{in} = 0~5.25V	-2.5	–	2.5	μA
Power Dissipation		P_D****		–	0.7	1.2	W
Input Capacitance	D_0~D_7	C_{in}	V_{in}=0V, T_a=25°C, f=1.0MHz	–	10	12.5	pF
	Except D_0~D_7			–	6.5	10	
Output Capacitance	A_0~A_{15}, R/$\overline{W}$, BA, VMA	C_{out}	V_{in}=0V, T_a=25°C, f=1.0MHz	–	–	12	pF

* T_a=25°C, V_{CC}=5V

** As $\overline{RES}$ input has histeresis character, applied voltage up to 2.4V is regarded as "Low" level when it goes up from 0V.

*** Does not include EXTAL and XTAL, which are crystal inputs.

**** In power-down mode, maximum power dissipation is less than 42mW.

● AC CHARACTERISTICS (V_{CC}=5.0V±5%, V_{CC} Standby=5.0V±5%, V_{SS}=0V, Ta=-20~+75°C, unless otherwise noted.)

1. CLOCK TIMING CHARACTERISTICS

Item		Symbol	Test Condition	min	typ	max	Unit
Frequency of Operation	Input Clock ÷ 4	f		0.1	–	1.0	MHz
	Crystal Frequency	f_{XTAL}		1.0	–	4.0	
Cycle Time		t_{cyc}	Fig. 4, Fig. 5	1.0	–	10	μs
Clock Pulse Width	"High" Level	$PW_{\phi H}$	at 2.4V (Fig. 4, Fig. 5)	450	–	4500	ns
	"Low" Level	$PW_{\phi L}$	at 0.8V (Fig. 4, Fig. 5)				
Clock Fall Time		t_{ϕ}	0.8V ~ 2.4V (Fig.4,Fig.5)	–	–	25	ns

2. READ/WRITE TIMING

Item	Symbol	Test Condition	min	typ*	max	Unit
Address Delay	t_{AD}	Fig. 4, Fig. 5, Fig. 8	–	–	270	ns
Peripheral Read Access Time	t_{acc}	Fig. 4	530	–	–	ns
Data Setup Time (Read)	t_{DSR}	Fig. 4	100	–	–	ns
Input Data Hold Time	t_{H}	Fig. 4	10	–	–	ns
Output Data Hold Time	t_{H}	Fig. 5	20	–	–	ns
Address Hold Time (Address, R/$\overline{W}$, VMA)	t_{AH}	Fig. 4, Fig. 5	10	–	–	ns
Data Delay Time (Write)	t_{DDW}	Fig. 5	–	–	225	ns
Bus Available Delay	t_{BA}	Fig. 6, Fig. 7, Fig. 9, Fig. 10	–	–	250	ns
Processor Controls Processor Control Setup Time	t_{PCS}	Fig. 6 ~ Fig. 9, Fig. 11	200	–	–	ns
Processor Control Rise and Fall Time (Measured at 0.8V and 2.0V)	t_{PCr}, t_{PCf}	Fig. 6 ~ Fig. 9, Fig. 11, Fig. 12, Fig. 14	–	–	100	ns

* Ta = 25°C, V_{CC} = 5V

3. POWER DOWN SEQUENCE TIMING, POWER UP RESET TIMING AND MEMORY READY TIMING

Item	Symbol	Test Condition	min	typ	max	Unit
RAM Enable Reset Time (1)	t_{RE1}	Fig. 12	150	–	–	ns
RAM Enable Reset Time (2)	t_{RE2}	Fig. 12	E-3 cycles	–	–	
Reset Release Time	t_{LRES}	Fig. 11	20	–	–	ms
RAM Enable Reset Time (3)	t_{RE3}	Fig. 11	0	–	–	ns
Memory Ready Setup Time	t_{SMR}	Fig. 14	300	–	–	ns
Memory Ready Hold Time	t_{HMR}	Fig. 14	0	–	200	ns

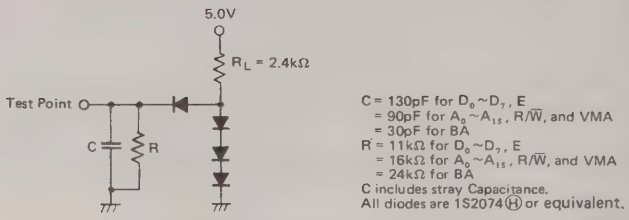

5.0V
R_L = 2.4kΩ
Test Point
C
R
C = 130pF for D_0~D_7, E
= 90pF for A_0~A_{15}, R/W, and VMA
= 30pF for BA
R = 11kΩ for D_0~D_7, E
= 16kΩ for A_0~A_{15}, R/W, and VMA
= 24kΩ for BA
C includes stray Capacitance.
All diodes are 1S2074Ⓗ or equivalent.

Figure 3 Bus Timing Test Load

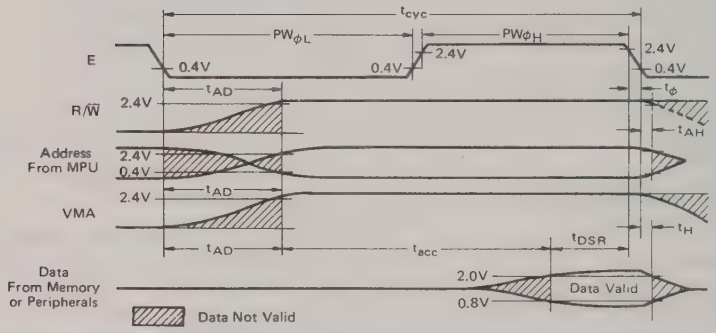


Figure 4 Read Data from Memory or Peripherals

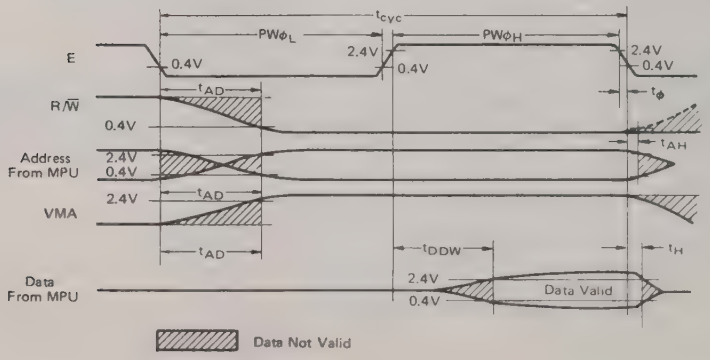


Figure 5 Write Data in Memory or Peripherals

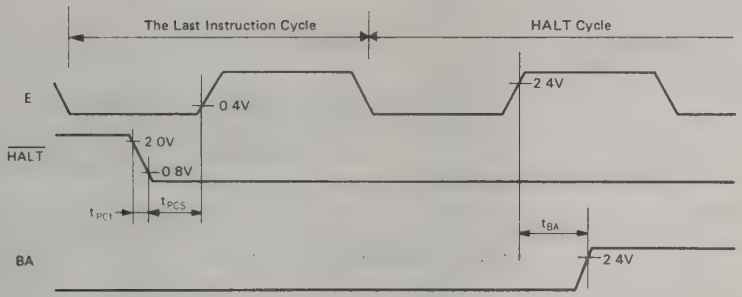


Figure 6 Timing of $\overline{\text{HALT}}$ and BA

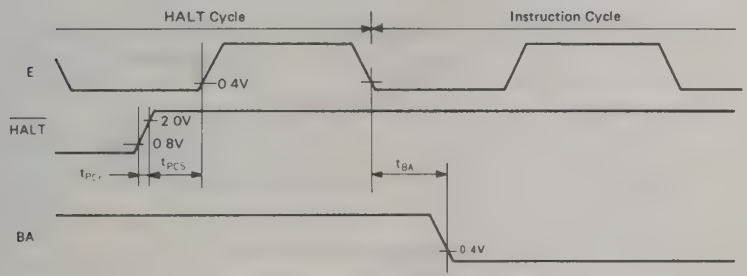


Figure 7 Timing of $\overline{\text{HALT}}$ and BA

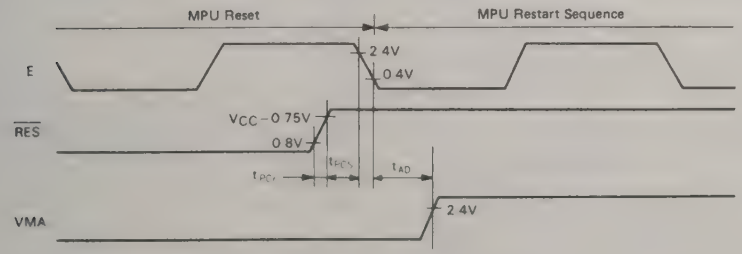


Figure 8 $\overline{\text{RES}}$ and MPU Restart Sequence

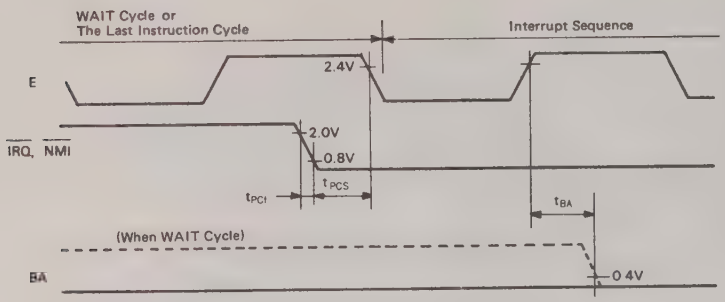


Figure 9 $\overline{IRQ}$ and $\overline{NMI}$ Interrupt Timing

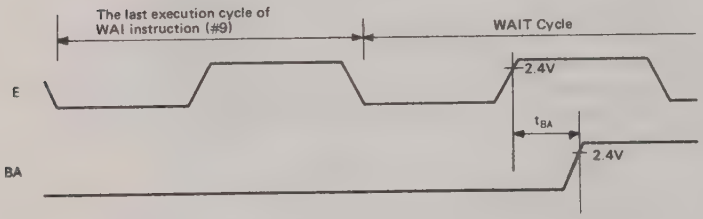


Figure 10 WAI Instruction and BA Timing

■ HD6802W MPU SIGNAL DESCRIPTION

- **Address Bus ($A_0 \sim A_{15}$)**

Sixteen pins are used for the address bus. The outputs are capable of driving one standard TTL load and 90pF.

- **Data Bus ($D_0 \sim D_7$)**

Eight pins are used for the data bus. It is bidirectional, transferring data to and from the memory and peripheral devices. It also has three-state output buffers capable of driving one standard TTL load and 130pF.

Data Bus will be in the output mode when the internal RAM is accessed. This prohibits external data entering the MPU. It should be noted that the internal RAM is fully decoded from $0000 to $00FF. External RAM at $0000 to $00FF must be disabled when internal RAM is accessed.

- **HALT**

When this input is in the "Low" state, all activity in the machine will be halted: This input is level sensitive.

In the halt mode, the machine will stop at the end of an instruction. Bus Available will be at a "High" state. Valid Memory Address will be at a "Low" state. The address bus will display the address of the next instruction.

To insure single instruction operation, transition of the $\overline{HALT}$ line must not occur during the last t_{PCS} of E and the $\overline{HALT}$ line must go "High" for one Clock cycle.

$\overline{HALT}$ should be tied "High" if not used. This is good engineering design practice in general and necessary to insure proper operation of the part.

- **Read/Write (R/$\overline{W}$)**

This TTL compatible output signals the peripherals and memory devices whether the MPU is in a Read ("High") or Write ("Low") state. The normal standby state of this signal is Read ("High"). When the processor is halted, it will be in the logical one state ("High").

This output is capable of driving one standard TTL load and 90pF.

- **Valid Memory Address (VMA)**

This output indicates to peripheral devices that there is a valid address on the address bus. In normal operation, this signal should be utilized for enabling peripheral interfaces such as the PIA and ACIA. This signal is not three-state. One standard TTL load and 90pF may be directly driven by this active high signal.

- **Bus Available (BA)**

The Bus Available signal will normally be in the "Low" state. When activated, it will go to the "High" state indicating that the microprocessor has stopped and that the address bus is available (but not in a three-state condition). This will occur if the $\overline{HALT}$ line is in the "Low" state or the processor is in the wait state as a result of the execution of a WAI instruction. At such time, all three-state output drivers will go to their off state and other outputs to their normally inactive level.

The processor is removed from the wait state by the occurrence of a maskable (mask bit I=0) or nonmaskable interrupt. This output is capable of driving one standard TTL load and 30pF.

- **Interrupt Request ($\overline{IRQ}$)**

This level sensitive input requests that an interrupt sequence be generated within the machine. The processor will wait, until it completes the current instruction that is being executed before it recognizes the request. At that time, if the interrupt mask bit in the Condition Code Register is not set, the machine will begin an interrupt sequence. The index Register, Program Counter, Accumulators, and Condition Code Register are stored away on the stack. Next the MPU will respond to the interrupt request by setting the interrupt mask bit high so that no further interrupts may occur. At the end of the cycle, a 16-bit address will be loaded that points to a vectoring address which is located in memory locations FFF8 and FFF9. An address loaded at these locations causes the MPU to branch to an interrupt routine in memory.

The $\overline{HALT}$ line must be in the "High" state for interrupts to be serviced. Interrupts will be latched internally while $\overline{HALT}$ is "Low".

A 3kΩ external register to V_{CC} should be used for wire-OR and optimum control of interrupts.

- **Reset ($\overline{RES}$)**

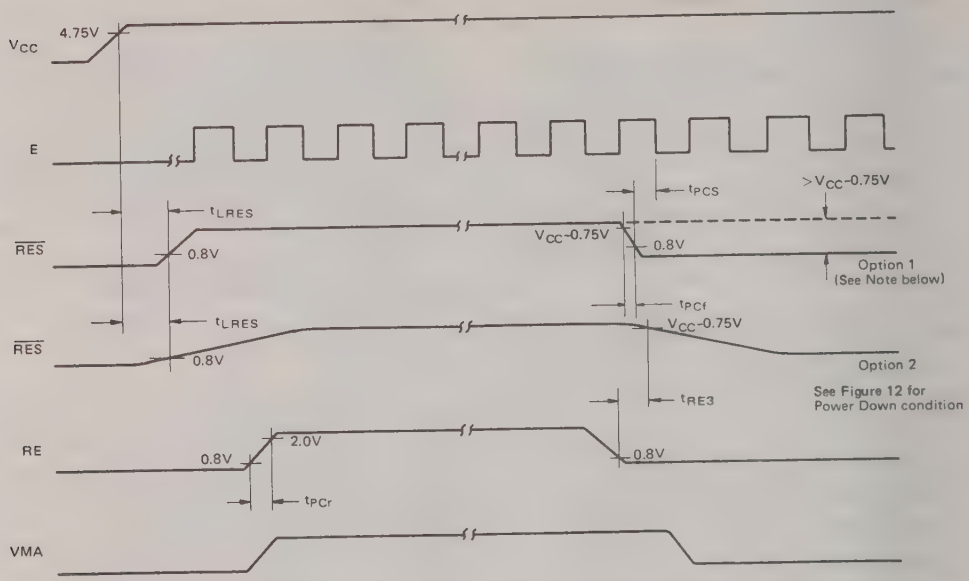
This input is used to reset and start the MPU from a power-down condition, resulting from a power failure or an initial start-up of the processor. When this line is "Low", the MPU is inactive and the information in the registers will be lost. If a "High" level is detected on the input, this will signal the MPU to begin the restart sequence. This will start execution of a routine to initialize the processor from its reset condition. All the higher order address lines will be forced "High". For the restart, the last two(FFFE, FFFF) locations in memory will be used to load the program that is addressed by the program counter. During the restart routine, the interrupt mask bit is set and must be reset before the MPU can be interrupted by $\overline{IRQ}$. Power-up and reset timing and power-down sequences are shown in Fig. 11 and Fig. 12 respectively.

- **Non-Maskable Interrupt ($\overline{NMI}$)**

A low-going edge on this input requests that a non-mask-interrupt sequence be generated within the processor. As with the $\overline{IRQ}$ signal, the processor will complete the current instruction that is being executed before it recognizes the $\overline{NMI}$ signal. The interrupt mask bit in the Condition Code Register has no effect on $\overline{NMI}$.

The Index Register, Program Counter, Accumulators, and Condition Code Register are stored away on the stack. At the end of the cycle, a 16-bit address will be loaded that points to a vectoring address which is located in memory locations FFFC and FFFD. An address loaded at these locations causes the MPU to branch to a non-maskable interrupt routine in memory. A 3kΩ external resistor to V_{CC} should be used for wire-OR and optimum control of interrupts.

Inputs $\overline{IRQ}$ and $\overline{NMI}$ are hardware interrupt lines that are sampled when E is "High" and will start the interrupt routine on a "Low" E following the completion of an instruction. $\overline{IRQ}$ and $\overline{NMI}$ should be tied "High" if not used. This is good engineering design practice in general and necessary to insure proper operation of the part. Fig. 13 is a flowchart describing the major decision paths and interrupt vectors of the microprocessor. Table 1 gives the memory map for interrupt vectors.



(NOTE) If option 1 is chosen, $\overline{RES}$ and RE pins can be tied together.

Figure 11 Power-up and Reset Timing

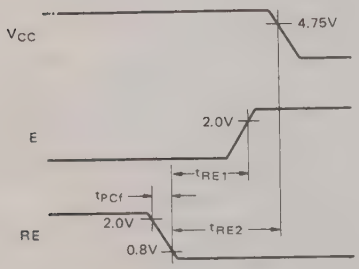


Figure 12 Power-down Sequence

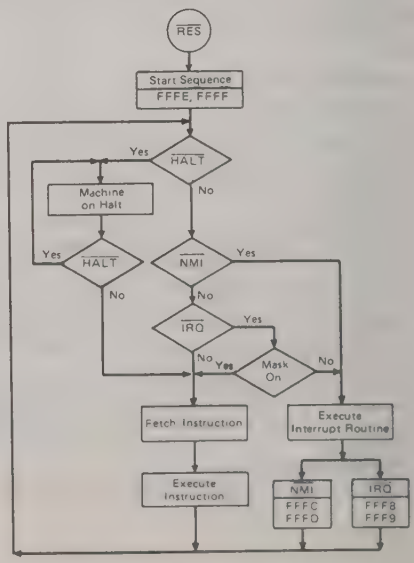


Figure 13 MPU Flow Chart

Table 1 Memory Map for Interrupt Vectors

Vector MS	Vector LS	Description	
FFFE	FFFF	Restart	($\overline{RES}$)
FFFC	FFFD	Non-Maskable Interrupt	($\overline{NMI}$)
FFFA	FFFB	Software Interrupt	(SWI)
FFF8	FFF9	Interrupt Request	($\overline{IRQ}$)

- **RAM Enable (RE)**

A TTL-compatible RAM enable input controls the on-chip RAM of the HD6802W. When placed in the "High" state, the on-chip memory is enabled to respond to the MPU controls. In the "Low" state, RAM is disabled. This pin may also be utilized to disable reading and writing the on-chip RAM during a power-down situation. RAM enable must be "Low" three cycles before V_{CC} goes below 4.75V during power-down.

RE should be tied to the correct "High" or "Low" state if not used. This is good engineering design practice in general and necessary to insure proper operation of the part.

- **EXTAL and XTAL**

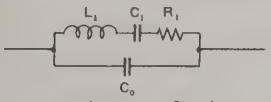
The HD6802W has an internal oscillator that may be crystal controlled. These connections are for a parallel resonant fundamental crystal (AT cut). A divide-by-four circuit has been added to the HD6802W so that a 4MHz crystal may be used in lieu of a 1MHz crystal for a more cost-effective system. Pin39 of the HD6802W may be driven externally by a TTL input signal if a separate clock is required. Pin38 is to be left open in this mode.

An RC network is not directly usable as a frequency source on pins 38 and 39. An RC network type TTL or CMOS oscillator will work well as long as the TTL or CMOS output drives the HD6802W.

If an external clock is used, it may not be halted for more than 4.5μs. The HD6802W is a dynamic part except for the internal RAM, and requires the external clock to retain information.

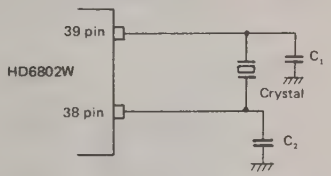
Conditions for Crystal (4 MHz)

- AT Cut Parallel resonant
- C_0 = 7 pF max.
- R_1 = 80Ω max.



Crystal Equivalent Circuit

Recommended Oscillator (4MHz)



$C_1 = C_2$ = 22pF ± 20%

When using the crystal, see the note for Board Design of the Oscillation Circuit in HD6802W.

- **Memory Ready (MR)**

MR is a TTL compatible input control signal which allows stretching of E. When MR is "High", E will be in normal operation. When MR is "Low", E may be stretched integral multiples of half periods, thus allowing interface to slow memories. Memory Ready timing is shown in Fig. 14.

MR should be tied "High" if not used. This is good engineering design practice in general and necessary to insure proper operation of the part. A maximum stretch is 4.5μs.

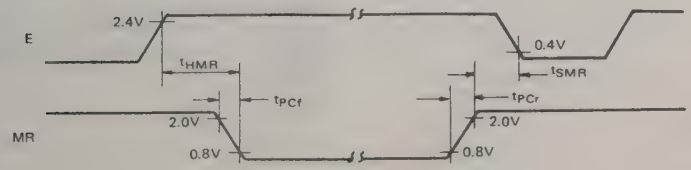


Figure 14 Memory Ready Control Function

- **Enable (E)**

This pin supplies the clock for the MPU and the rest of the system. This is a single phase, TTL compatible clock. This clock may be conditioned by a Memory Ready Signal. This is equivalent to ϕ_2 on the HD6800.

- **V_{CC} Standby**

This pin supplies the dc voltage to the first 32 bytes of RAM as well as the RAM Enable (RE) control logic. Thus retention of data in this portion of the RAM on a power-up, power-down, or standby condition is guaranteed at the range of 4.0 V to 5.25 V.

Maximum current drain at 5.25V is 8mA.

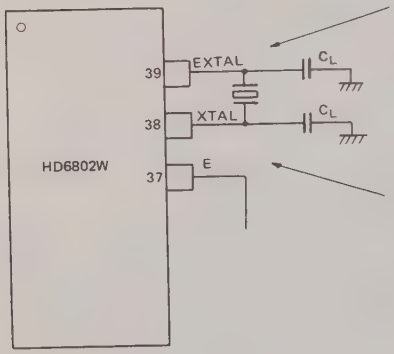
■ MPU INSTRUCTION SET

The HD6802W has a set of 72 different instructions. Included are binary and decimal arithmetic, logical, shift, rotate, load, store, conditional or unconditional branch, interrupt and stack manipulation instructions.

This instruction set is the same as that for the 6800MPU (HD6800 etc.) and is not explained again in this data sheet.

■ NOTE FOR BOARD DESIGN OF THE OSCILLATION CIRCUIT IN HD6802W

In designing the board, the following notes should be taken when the crystal oscillator is used.

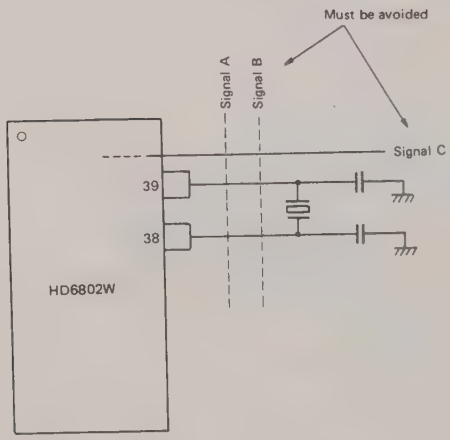


Crystal oscillator and load capacity C_L must be placed near the LSI as much as possible.

Normal oscillation may be disturbed when external noise is induced to pin 38 and 39.

Pin 38 signal line should be wired apart from pin 37 signal line as much as possible. Don't wire them in parallel, or normal oscillation may be disturbed when E signal is feedbacked to XTAL.

The following design must be avoided.



A signal line or a power source line must not cross or go near the oscillation circuit line as shown in the left figure to prevent the induction from these lines and perform the correct oscillation. The resistance among XTAL, EXTAL and other pins should be over 10MΩ.

Figure 15 Note for Board Design of the Oscillation Circuit

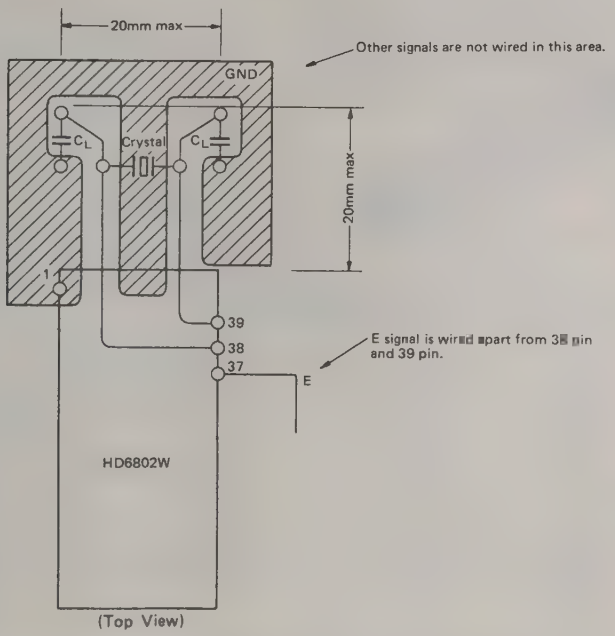


Figure 16 Example of Board Design Using the Crystal Oscillator

■ NOTE FOR THE RELATION BETWEEN WAI INSTRUCTION AND HALT OPERATION OF HD6802W

When $\overline{HALT}$ input signal is asserted to "Low" level, the MPU will be halted after the execution of the current instruction except WAI instruction.

The "Halt" signal is not accepted after the fetch cycle of the WAI instruction (See ① in Fig. 17). In the case of the "WAI" instruction, the MPU enters the "WAIT" cycle after stacking the internal registers and outputs the "High" level on the BA line.

When an interrupt request signal is input to the MPU, the MPU accepts the interrupt regardless the "Halt" signal and releases the "WAIT" state and outputs the interrupt's vector address. If the "Halt" signal is "Low" level, the MPU halts after the fetch of new PC contents. The sequense is shown below.

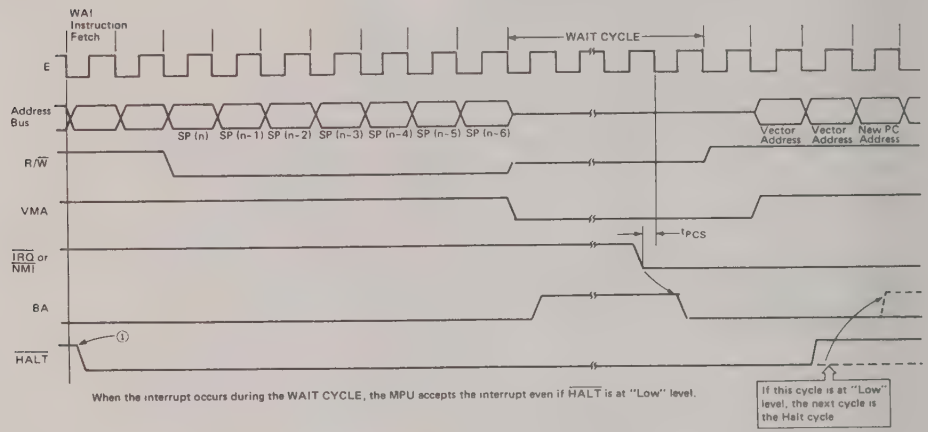

WAI Instruction Fetch
WAIT CYCLE
E
Address Bus
SP (n) SP (n-1) SP (n-2) SP (n-3) SP (n-4) SP (n-5) SP (n-6)
Vector Address
Vector Address
New PC Address
R/W
VMA
tPCS
IRQ or NMI
BA
HALT
①
When the interrupt occurs during the WAIT CYCLE, the MPU accepts the interrupt even if HALT is at "Low" level.
If this cycle is at "Low" level, the next cycle is the Halt cycle

Figure 17 HD6802W WAIT CYCLE & $\overline{HALT}$ Request

HD6809,HD68A09,HD68B09
MPU (Micro Processing Unit)

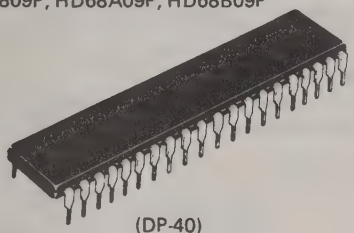
The HD6809 is a revolutionary high performance 8-bit microprocessor which supports modern programming techniques such as position independence, reentrancy, and modular programming.

This third-generation addition to the HMCS6800 family has major architectural improvements which include additional registers, instructions and addressing modes.

The basic instructions of any computer are greatly enhanced by the presence of powerful addressing modes. The HD6809 has the most complete set of addressing modes available on any 8-bit microprocessor today.

The HD6809 has hardware and software features which make it an ideal processor for higher level language execution or standard controller applications.

HD6809P, HD68A09P, HD68B09P



(DP-40)

HD6800 COMPATIBLE

- Hardware — Interfaces with All HMCS6800 Peripherals
- Software — Upward Source Code Compatible Instruction Set and Addressing Modes

■ ARCHITECTURAL FEATURES

- Two 16-bit Index Registers
- Two 16-bit Indexable Stack Pointers
- Two 8-bit Accumulators can be Concatenated to Form One 16-Bit Accumulator
- Direct Page Register Allows Direct Addressing Throughout Memory

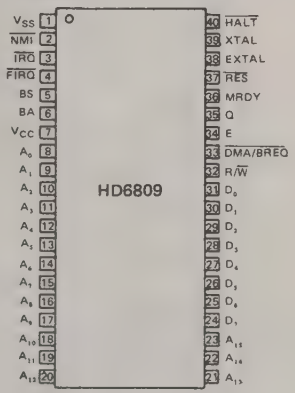
■ HARDWARE FEATURES

- On Chip Oscillator
- $\overline{DMA/BREQ}$ Allows DMA Operation or Memory Refresh
- Fast Interrupt Request Input Stacks Only Condition Code Register and Program Counter
- MRDY Input Extends Data Access Times for Use With Slow Memory
- Interrupt Acknowledge Output Allows Vectoring By Devices
- SYNC Acknowledge Output Allows for Synchronization to External Event
- Single Bus-Cycle RESET
- Single 5-Volt Supply Operation
- NMI Blocked After RESET Until After First Load of Stack Pointer
- Early Address Valid Allows Use With Slower Memories
- Early Write-Data for Dynamic Memories
- Compatible with MC6809, MC68A09 and MC68B09

■ SOFTWARE FEATURES

- 10 Addressing Modes
  - HMCS6800 Upward Compatible Addressing Modes
  - Direct Addressing Anywhere in Memory Map
  - Long Relative Branches
  - Program Counter Relative
  - True Indirect Addressing
  - Expanded Indexed Addressing:

■ PIN ARRANGEMENT

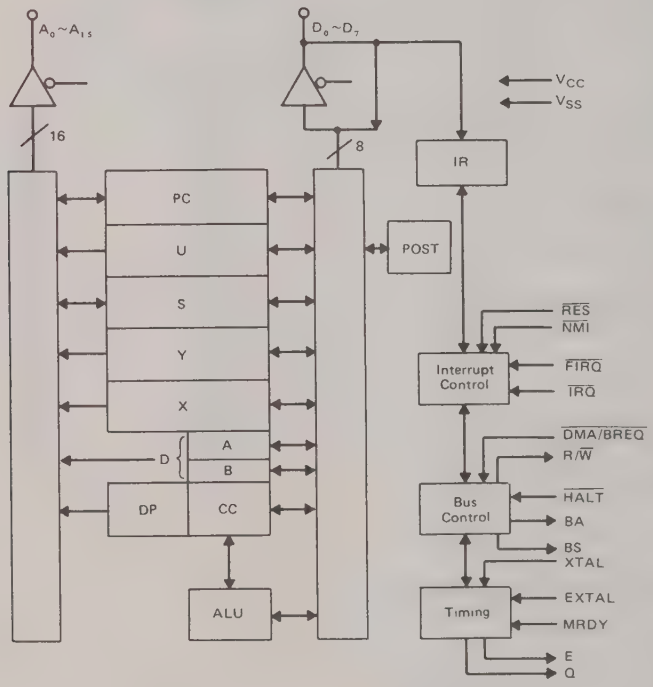


(Top View)

0, 5, 8, or 16-bit Constant Offsets
8, or 16-bit Accumulator Offsets
Auto-Increment/Decrement by 1 or 2

- Improved Stack Manipulation
- 1464 Instructions with Unique Addressing Modes
- 8 x 8 Unsigned Multiply
- 16-bit Arithmetic
- Transfer/Exchange All Registers
- Push/Pull Any Registers or Any Set of Registers
- Load Effective Address

■ BLOCK DIAGRAM



■ ABSOLUTE MAXIMUM RATINGS

Item	Symbol	Value	Unit
Supply Voltage	V_{CC}*	-0.3 ~ +7.0	V
Input Voltage	V_{in}*	-0.3 ~ +7.0	V
Operating Temperature	T_{opr}	-20 ~ +75	°C
Storage Temperature	T_{stg}	-55 ~ +150	°C

* With respect to V_{SS} (SYSTEM GND)

(NOTE) Permanent LSI damage may occur if maximum ratings are exceeded. Normal operation should be under recommended operating conditions. If these conditions are exceeded, it could affect reliability of LSI.

■ RECOMMENDED OPERATING CONDITIONS

Item	Symbol		min	typ	max	Unit
Supply Voltage	V_{CC}*		4.75	5.0	5.25	V
Input Voltage	V_{IL}*		-0.3	–	0.8	V
	V_{IH}*	Logic (Ta = 0 ~ +75°C)	2.0	–	V_{CC}	V
		Logic (Ta = -20 ~ 0°C)	2.2	–	V_{CC}	
		$\overline{RES}$	4.0	–	V_{CC}	
Operating Temperature	T_{opr}		-20	25	75	°C

* With respect to V_{SS} (SYSTEM GND)

■ ELECTRICAL CHARACTERISTICS

● DC CHARACTERISTICS (V_{CC}=5V±5%, V_{SS} = 0V, Ta = -20~+75°C, unless otherwise noted.)

Item		Symbol	Test Condition	HD6809			HD68A09			HD68B09			Unit
				min	typ*	max	min	typ*	max	min	typ*	max	
Input "High" Voltage	Except $\overline{RES}$	V_{IH}	Ta = 0 ~ +75°C	2.0	–	V_{CC}	2.0	–	V_{CC}	2.0	–	V_{CC}	V
			Ta = -20 ~ 0°C	2.2	–	V_{CC}	2.2	–	V_{CC}	2.2	–	V_{CC}	
	$\overline{RES}$			4.0	–	V_{CC}	4.0	–	V_{CC}	4.0	–	V_{CC}	
Input "Low" Voltage		V_{IL}		-0.3	–	0.8	-0.3	–	0.8	-0.3	–	0.8	V
Input Leakage Current	Except EXTAL, XTAL	I_{in}	Vin=0~5.25V, V_{CC}=max	-2.5	–	2.5	-2.5	–	2.5	-2.5	–	2.5	μA
Three State (Off State) Input Current	D_0~D_7	I_{TSI}	Vin=0.4~2.4V, V_{CC}=max	-10	–	10	-10	–	10	-10	–	10	μA
	A_0~A_{15}, R/$\overline{W}$			-100	–	100	-100	–	100	-100	–	100	
Output "High" Voltage	D_0~D_7	V_{OH}	I_{LOAD}=-205μA, V_{CC}=min	2.4	–	–	2.4	–	–	2.4	–	–	V
	A_0~A_{15}, R/$\overline{W}$, Q, E		I_{LOAD}=-145μA, V_{CC}=min	2.4	–	–	2.4	–	–	2.4	–	–	
	BA, BS		I_{LOAD}=-100μA, V_{CC}=min	2.4	–	–	2.4	–	–	2.4	–	–	
Output "Low" Voltage		V_{OL}	I_{LOAD}=2mA	–	–	0.5	–	–	0.5	–	–	0.5	V
Power Dissipation		P_D		–	–	1.0	–	–	1.0	–	–	1.0	W
Input Capacitance	D_0~D_7	Cin	Vin=0V, Ta=25°C, f=1MHz	–	10	15	–	10	15	–	10	15	pF
	Except D_0~D_7			–	7	10	–	7	10	–	7	10	
Output Capacitance	A_0~A_{15}, R/$\overline{W}$, BA, BS	Cout		–	–	12	–	–	12	–	–	12	pF

*Ta=25°C, V_{CC}=5V

- **AC CHARACTERISTICS (V_{CC}=5V±5%, V_{SS} = 0V, Ta = -20~+75°C, unless otherwise noted.)**

1. CLOCK TIMING

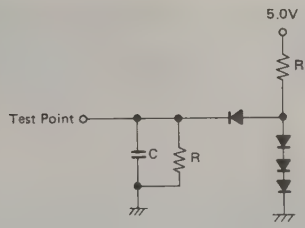
Item	Symbol	Test Condition	HD6809			HD68A09			HD68B09			Unit
			min	typ	max	min	typ	max	min	typ	max	
Frequency of Operation (Crystal or External Input)	f_{XTAL}	Fig. 2, Fig. 3	0.4	–	4	0.4	–	6	0.4	–	8	MHz
Cycle Time	t_{cyc}		1000	–	10000	667	–	10000	500	–	10000	ns
Total Up Time	t_{UT}		975	–	–	640	–	–	480	–	–	ns
Processor Clock "High"	t_{PWEH}		450	–	15500	280	–	15700	220	–	15700	ns
Processor Clock "Low"	t_{PWEL}		430	–	5000	280	–	5000	210	–	5000	ns
E Rise and Fall Time	t_{Er}, t_{Ef}		–	–	25	–	–	25	–	–	20	ns
E_{Low} to Q_{High} Time	t_{AVS}		200	–	250	130	–	165	80	–	125	ns
Q Clock "High"	t_{PWQH}		450	–	5000	280	–	5000	220	–	5000	ns
Q Clock "Low"	t_{PWQL}		450	–	15500	280	–	15700	220	–	15700	ns
Q Rise and Fall Time	t_{Qr}, t_{Qf}		–	–	25	–	–	25	–	–	20	ns
Q_{Low} to E Falling	t_{QE}		200	–	–	133	–	–	100	–	–	ns

2. BUS TIMING

Item	Symbol	Test Condition	HD6809			HD68A09			HD68B09			Unit
			min	typ	max	min	typ	max	min	typ	max	
Address Delay	t_{AD}	Fig. 2, Fig. 3	–	–	200	–	–	140	–	–	110	ns
Address Valid to Q_{High}	t_{AQ}		50	–	–	25	–	–	15	–	–	ns
Peripheral Read Access Time ($t_{UT}-t_{AD}-t_{DSR}=t_{ACC}$)	t_{ACC}		695	–	–	440	–	–	330	–	–	ns
Data Set Up Time (Read)	t_{DSR}		80	–	–	60	–	–	40	–	–	ns
Input Data Hold Time	t_{DHR}		10	–	–	10	–	–	10	–	–	ns
Address Hold Time (A_0~A_{15}, R/$\overline{W}$)	t_{AH}	Fig. 2, Fig. 3 Ta=0~+75°C	20	–	–	20	–	–	20	–	–	ns
		Fig. 2, Fig. 3 Ta=-20~0°C	10	–	–	10	–	–	10	–	–	ns
Data Delay Time (Write)	t_{DDW}	Fig. 3	–	–	200	–	–	140	–	–	110	ns
Output Hold Time	t_{DHW}	Fig. 3 Ta=0~+75°C	30	–	–	30	–	–	30	–	–	ns
		Fig. 3 Ta=-20~0°C	20	–	–	20	–	–	20	–	–	ns

3. PROCESSOR CONTROL TIMING

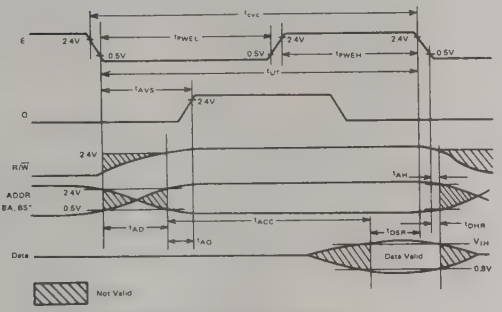
Item	Symbol	Test Condition	HD6809			HD68A09			HD68B09			Unit
			min	typ	max	min	typ	max	min	typ	max	
MRDY Set Up Time	t_{PCSM}	Fig. 6~Fig. 10 Fig. 14, Fig. 15	125	–	–	125	–	–	110	–	–	ns
Interrupts Set Up Time	t_{PCS}		200	–	–	140	–	–	110	–	–	ns
$\overline{HALT}$ Set Up Time	t_{PCSH}		200	–	–	140	–	–	110	–	–	ns
$\overline{RES}$ Set Up Time	t_{PCSR}		200	–	–	140	–	–	110	–	–	ns
$\overline{DMA/BREQ}$ Set Up Time	t_{PCSD}		125	–	–	125	–	–	110	–	–	ns
Processor Control Rise and Fall Time	t_{PCr}, t_{PCf}		–	–	100	–	–	100	–	–	100	ns
Crystal Oscillator Start Time	t_{RC}		–	–	50	–	–	30	–	–	30	ms



• C = 30pF (BA, BS)
130pF ($D_0 \sim D_7$, E, Q)
90pF ($A_0 \sim A_{15}$, R/$\overline{W}$)
• R = 11kΩ ($D_0 \sim D_7$)
16kΩ ($A_8 \sim A_{15}$, E, Q, R/$\overline{W}$)
24kΩ (BA, BS)

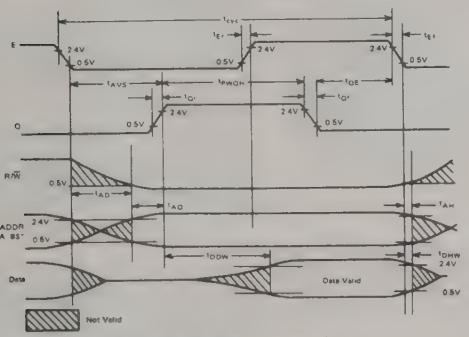
All diodes are 1S2074(H) or equivalent.
C includes Stray Capacitance.

Figure 1 Bus Timing Test Load



*Hold time for BA, BS not specified.

Figure 2 Read Data from Memory or Peripherals



*Hold time for BA, BS not specified.

Figure 3 Write Data to Memory or Peripherals

■ PROGRAMMING MODEL

As shown in Figure 4, the HD6809 adds three registers to the set available in the HD6800. The added registers include a Direct Page Register, the User Stack pointer and a second Index Register.

• Accumulators (A, B, D)

The A and B registers are general purpose accumulators which are used for arithmetic calculations and manipulation of data.

Certain instructions concatenate the A and B registers to form a single 16-bit accumulator. This is referred to as the D register, and is formed with the A register as the most significant byte.

• Direct Page Register (DP)

The Direct Page Register of the HD6809 serves to enhance the Direct Addressing Mode. The content of this register appears at the higher address outputs ($A_8 \sim A_{15}$) during Direct Addressing Instruction execution. This allows the direct mode to be used at any place in memory, under program control. To ensure HD6800 compatibility, all bits of this register are cleared during Processor Reset.

- **Index Registers (X, Y)**

The Index Registers are used in indexed mode of addressing. The 16-bit address in this register takes part in the calculation of effective addresses. This address may be used to point to data directly or may be modified by an optional constant or register offset. During some indexed modes, the contents of the index register are incremented or decremented to point to the next item of tabular type data. All four pointer registers (X, Y, U, S) may be used as index registers.

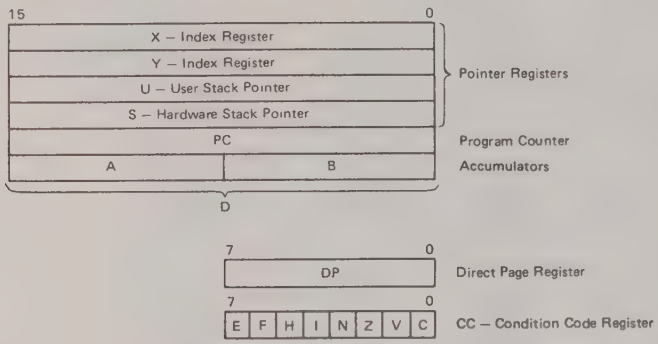


Figure 4 Programming Model of The Microprocessing Unit

- **Stack Pointer (U, S)**

The Hardware Stack Pointer (S) is used automatically by the processor during subroutine calls and interrupts. The stack pointers of the HD6809 point to the top of the stack, in contrast to the HD6800 stack pointer, which pointed to the next free location on the stack. The User Stack Pointer (U) is controlled exclusively by the programmer thus allowing arguments to be passed to and from subroutines with ease. Both Stack Pointers have the same indexed mode addressing capabilities as the X and Y registers, but also support Push and Pull instructions. This allows the HD6809 to be used efficiently as a stack processor, greatly enhancing its ability to support higher level languages and modular programming.

- **Program Counter**

The Program Counter is used by the processor to point to the address of the next instruction to be executed by the processor. Relative Addressing is provided allowing the Program Counter to be used like an index register in some situations.

- **Condition Code Register**

The Condition Code Register defines the State of the Processor at any given time. See Fig. 5.

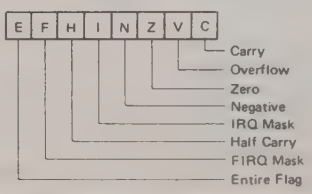


Figure 5 Condition Code Register Format

■ CONDITION CODE REGISTER DESCRIPTION

- **Bit 0 (C)**

Bit 0 is the carry flag, and is usually the carry from the binary ALU. C is also used to represent a 'borrow' from subtract like instructions (CMP, NEG, SUB, SBC) and is the complement of the carry from the binary ALU.

- **Bit 1 (V)**

Bit 1 is the overflow flag, and is set to a one by an operation which causes a signed two's complement arithmetic overflow. This overflow is detected in an operation in which the carry from the MSB in the ALU does not match the carry from the MSB-1.

- **Bit 2 (Z)**

Bit 2 is the zero flag, and is set to a one if the result of the previous operation was identically zero.

- **Bit 3 (N)**

Bit 3 is the negative flag, which contains exactly the value of the MSB of the result of the preceding operation. Thus, a negative two's-complement result will leave N set to a one.

- **Bit 4 (I)**

Bit 4 is the $\overline{IRQ}$ mask bit. The processor will not recognize interrupts from the $\overline{IRQ}$ line if this bit is set to a one. $\overline{NMI}$, $\overline{FIRQ}$, $\overline{IRQ}$, $\overline{RES}$, and SWI all are set I to a one; SWI2 and SWI3 do not affect I.

- **Bit 5 (H)**

Bit 5 is the half-carry bit, and is used to indicate a carry from bit 3 in the ALU as a result of an 8-bit addition only (ADC or ADD). This bit is used by the DAA instruction to perform a BCD decimal add adjust operation. The state of this flag is

undefined in all subtract-like instructions.

- **Bit 6 (F)**

Bit 6 is the $\overline{FIRQ}$ mask bit. The processor will not recognize interrupts from the $\overline{FIRQ}$ line if this bit is a one. $\overline{NMI}$, $\overline{FIRQ}$, SWI, and $\overline{RES}$ all set F to a one. $\overline{IRQ}$, SWI2 and SWI3 do not affect F.

- **Bit 7 (E)**

Bit 7 is the entire flag, and when set to a one indicates that the complete machine state (all the registers) was stacked, as opposed to the subset state (PC and CC). The E bit of the stacked CC is used on a return from interrupt (RTI) to determine the extent of the unstacking. Therefore, the current E left in the Condition Code Register represents past action.

■ SIGNAL DESCRIPTION

- **Power (V_{SS}, V_{CC})**

Two pins are used to supply power to the part: V_{SS} is ground or 0 volts, while V_{CC} is +5.0V ±5%.

- **Address Bus (A_0~A_{15})**

Sixteen pins are used to output address information from the MPU onto the Address Bus. When the processor does not require the bus for a data transfer, it will output address $FFFF_{16}$, R/$\overline{W}$ = "High", and BS = "Low"; this is a "dummy access" or $\overline{VMA}$ cycle. Addresses are valid on the rising edge of Q (see Figs. 2 and 3). All address bus drivers are made high impedance when output Bus Availalbe (BA) is "High". Each pin will drive one Schottky TTL load or four LS TTL loads, and typically 90 pF.

- **Data Bus (D_0~D_7)**

These eight pins provide communication with the system bi-directional data bus. Each pin will drive one Schottky TTL load or four LS TTL loads, and typically 130 pF.

- **Read/Write (R/$\overline{W}$)**

This signal indicates the direction of data transfer on the data bus. A "Low" indicates that the MPU is writing data onto the data bus. R/$\overline{W}$ is made high impedance when BA is "High". R/$\overline{W}$ is valid on the rising edge of Q. Refer to Figs. 2 and 3.

- **Reset ($\overline{RES}$)**

A "Low" level on this Schmitt-trigger input for greater than one bus cycle will reset the MPU, as shown in Fig. 6. The Reset vectors are fetched from locations $FFFE_{16}$ and $FFFF_{16}$ (Table 1) when Interrupt Acknowledge is true, ($\overline{BA}$ · BS=1). During initial power-on, the Reset line should be held "Low" until the clock oscillator is fully operational. See Fig. 7.

Because the HD6809 Reset pin has a Schmitt-trigger input with a threshold voltage higher than that of standard peripherals, a simple R/C network may be used to reset the entire system. This higher threshold voltage ensures that all peripherals are out of the reset state before the Processor.

Table 1 Memory Map for Interrupt Vectors

Memory Map For Vector Locations		Interrupt Vector Description
MS	LS	
FFFE	FFFF	$\overline{RES}$
FFFC	FFFD	$\overline{NMI}$
FFFA	FFFB	SWI
FFF8	FFF9	$\overline{IRQ}$
FFF6	FFF7	$\overline{FIRQ}$
FFF4	FFF5	SWI2
FFF2	FFF3	SWI3
FFF0	FFF1	Reserved

- **$\overline{HALT}$**

A "Low" level on this input pin will cause the MPU to stop running at the end of the present instruction and remain halted indefinitely without loss of data. When halted, the BA output is driven "High" indicating the buses are high impedance. BS is also "High" which indicates the processor is in the Halt or Bus Grant state. While halted, the MPU will not respond to external real-time requests ($\overline{FIRQ}$, $\overline{IRQ}$) although $\overline{DMA/BREQ}$ will always be accepted, and $\overline{NMI}$ or $\overline{RES}$ will be latched for later response. During the Halt state Q and E continue to run normally. If the MPU is not running ($\overline{RES}$, $\overline{DMA/BREQ}$), a halted state (BA · BS=1) can be achieved by pulling $\overline{HALT}$ "Low" while $\overline{RES}$ is still "Low". If $\overline{DAM/BREQ}$ and $\overline{HALT}$ are both pulled "Low", the processor will reach the last cycle of the instruction (by reverse cycle stealing) where the machine will then become halted. See Figs. 8 and 16.

- **Bus Available, Bus Status (BA, BS)**

The BA output is an indication of an internal control signal which makes the MOS buses of the MPU high impedance. This signal does not imply that the bus will be available for more than one cycle. When BA goes "Low", an additional dead cycle will elapse before the MPU acquires the bus.

The BS output signal, when decoded with BA, represents the MPU state (valid with leading edge of Q).

Table 2 MPU State Definition

BA	BS	MPU State
0	0	Normal (Running)
0	1	Interrupt or RESET Acknowledge
1	0	SYNC Acknowledge
1	1	HALT or Bus Grant

Interrupt Acknowledge is indicated during both cycles of a hardware-vector-fetch ($\overline{RES}$, $\overline{NMI}$, $\overline{FIRQ}$, $\overline{IRQ}$, SWI, SWI2, SWI3). This signal, plus decoding of the lower four address lines, can provide the user with an indication of which interrupt level is being serviced and allow vectoring by device. See Table 1.

Sync Acknowledge is indicated while the MPU is waiting for external synchronization on an interrupt line.

Halt/Bus Grant is true when the HD6809 is in a Halt or Bus Grant condition.

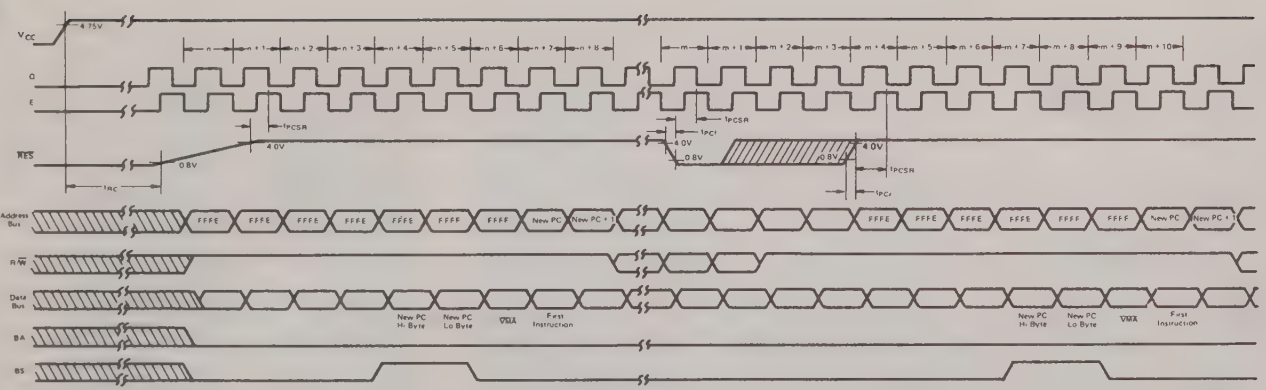
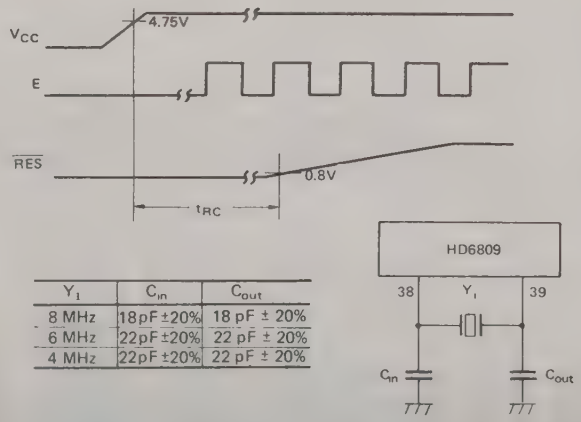


Figure 6 $\overline{\text{RES}}$ Timing



Y_1	C_{in}	C_{out}
8 MHz	18pF ±20%	18 pF ± 20%
6 MHz	22pF ±20%	22 pF ± 20%
4 MHz	22pF ±20%	22 pF ± 20%

Figure 7 Crystal Connections and Oscillator Start Up

HITACHI

- **Non Maskable Interrupt ($\overline{NMI}$)***

A negative edge on this input requests that a non-maskable interrupt sequence be generated. A non-maskable interrupt cannot be inhibited by the program, and also has a higher priority than $\overline{FIRQ}$, $\overline{IRQ}$ or software interrupts. During recognition of an $\overline{NMI}$, the entire machine state is saved on the hardware stack. After reset, an $\overline{NMI}$ will not be recognized until the first program load of the Hardware Stack Pointer (S). The pulse width of $\overline{NMI}$ "Low" must be at least one E cycle. If the $\overline{NMI}$ input does not meet the minimum set up with respect to Q, the interrupt will not be recognized until the next cycle. See Fig. 9.

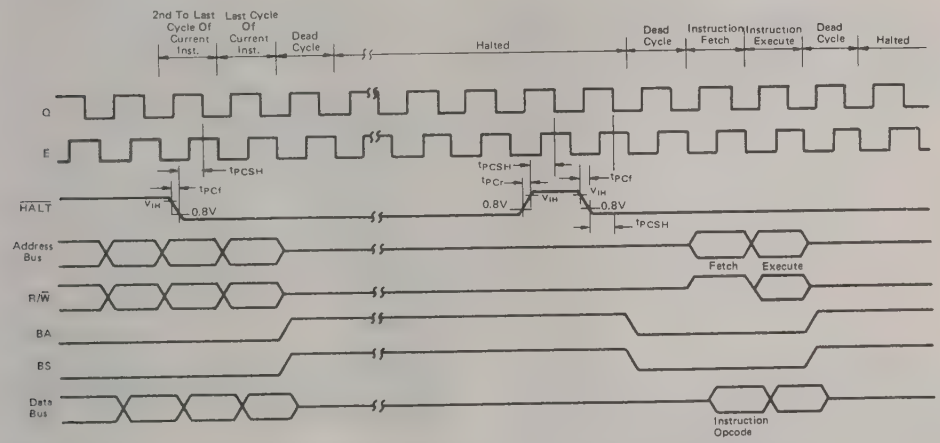


Figure 8 $\overline{HALT}$ and Single Instruction Execution for System Debug

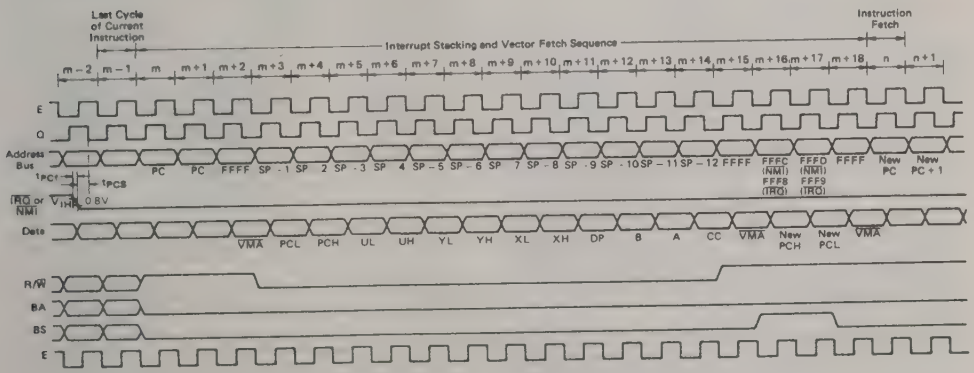


Figure 9 $\overline{IRQ}$ and $\overline{NMI}$ Interrupt Timing

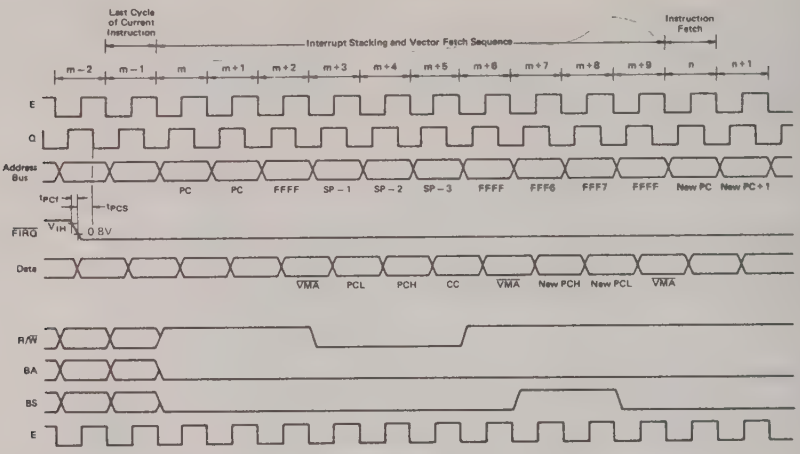


Figure 10 $\overline{\text{FIRQ}}$ Interrupt Timing

- **Fast-Interrupt Request ($\overline{\text{FIRQ}}$)***

A "Low" level on this input pin will initiate a fast interrupt sequence provided its mask bit (F) in the CC is clear. This sequence has priority over the standard Interrupt Request ($\overline{\text{IRQ}}$), and is fast in the sense that it stacks only the contents of the condition code register and the program counter. The interrupt service routine should clear the source of the interrupt before doing an RTI. See Fig. 10.

- **Interrupt Request ($\overline{\text{IRQ}}$)***

A "Low" level input on this pin will initiate an interrupt Request sequence provided the mask bit (I) in the CC is clear. Since $\overline{\text{IRQ}}$ stacks the entire machine state it provides a slower response to interrupts than $\overline{\text{FIRQ}}$. $\overline{\text{IRQ}}$ also has a lower priority than $\overline{\text{FIRQ}}$. Again, the interrupt service routine should clear the source of the interrupt before doing an RTI. See Fig. 9.

* $\overline{\text{NMI}}$, $\overline{\text{FIRQ}}$, and $\overline{\text{IRQ}}$ requests are sampled on the falling edge of Q. One cycle is required for synchronization before these interrupts are recognized. The pending interrupt(s) will not be serviced until completion of the current instruction unless a SYNC or CWAI condition is present. If $\overline{\text{IRQ}}$ and $\overline{\text{FIRQ}}$ do not remain "Low" until completion of the current instruction they may not be recognized. However, $\overline{\text{NMI}}$ is latched and need only remain "Low" for one cycle.

- **XTAL, EXTAL**

These inputs are used to connect the on-chip oscillator to an external parallel-resonant crystal. Alternately, the pin EXTAL may be used as a TTL level input for external timing by grounding XTAL. The crystal or external frequency is four times the bus frequency. See Fig. 7. Proper RF layout techniques should be observed in the layout of printed circuit boards.

<NOTE FOR BOARD DESIGN OF THE OSCILLATION CIRCUIT>

In designing the board, the following notes should be taken when the crystal oscillator is used.

1) Crystal oscillator and load capacity Cin, Cout must be placed near the LSI as much as possible.

Normal oscillation may be disturbed when external noise is induced to pin 38 and 39.

2) Pin 38 and 39 signal line should be wired apart from other signal line as much as possible. Don't wire them in parallel.

Normal oscillation may be disturbed when E or Q signal is feedbacked to pin 38 and 39.

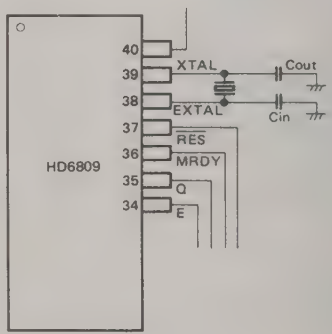


Figure 11 Board Design of the Oscillation Circuit.

<THE FOLLOWING DESIGN MUST BE AVOIDED>

A signal line or a power source line must not cross or go near the oscillation circuit line as shown in Fig. 12 to prevent the induction from these lines and perform the correct oscillation. The resistance among XTAL, EXTAL and other pins should be over 10MΩ.

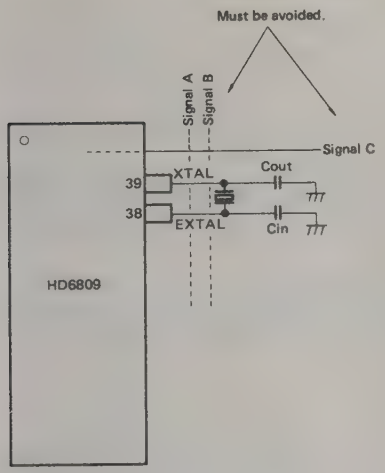


Figure 12 Example of Normal Oscillation may be Disturbed.

- **E, Q**

E is similar to the HD6800 bus timing signal ϕ_2; Q is a quadrature clock signal which leads E. Q has no parallel on the HD6800. Addresses from the MPU will be valid with the leading edge of Q. Data is latched on the falling edge of E. Timing for E and Q is shown in Fig. 13.

- **MRDY**

This input control signal allows stretching of E and Q to extend data-access time. E and Q operate normally while MRDY is "High". When MRDY is "Low", E and Q may be stretched in integral multiples of quarter (1/4) bus cycles, thus allowing interface to slow memories, as shown in Fig. 14. A maximum

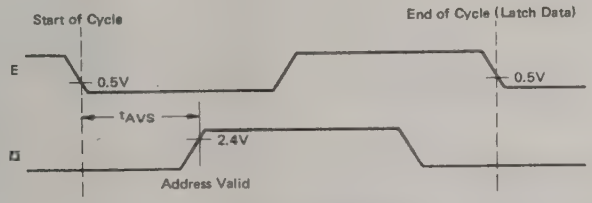


Figure 13 E/Q Relationship

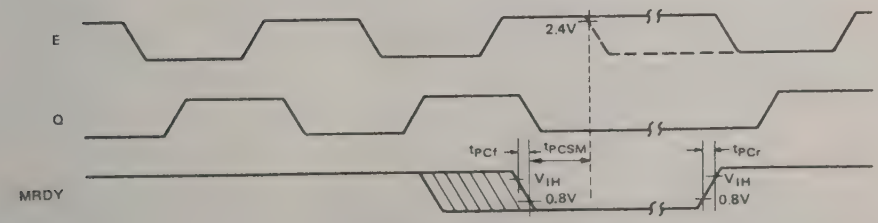


Figure 14 MRDY Timing

stretch is 10 microseconds. During nonvalid memory access ($\overline{VMA}$ cycles) MRDY has no effect on stretching E and Q; this inhibits slowing the processor during "don't care" bus accesses. MRDY may also be used to stretch clocks (for slow memory) when bus control has been transferred to an external device (through the use of $\overline{HALT}$ and $\overline{DMA/BREQ}$).
Also MRDY has effect on stretching E and Q during Dead Cycle.

- **$\overline{DMA/BREQ}$**

The $\overline{DMA/BREQ}$ input provides a method of suspending execution and acquiring the MPU bus for another use, as shown in Fig. 15. Typical uses include DMA and dynamic memory refresh.

Transition of $\overline{DMA/BREQ}$ should occur during Q. A "Low" level on this pin will stop instruction execution at the end of the current cycle. The MPU will acknowledge $\overline{DMA/BREQ}$ by setting BA and BS to "High" level. The requesting device will now have up to 15 bus cycles before the MPU retrieves the bus for self-refresh. Self-refresh requires one bus cycle with a leading and trailing dead cycle. See Fig. 16.

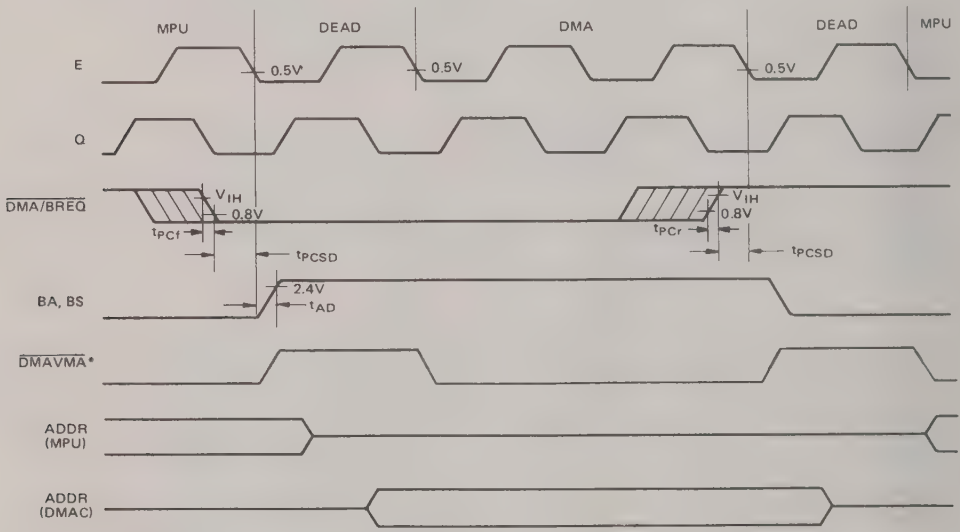
Typically, the DMA controller will request to use the bus by asserting $\overline{DMA/BREQ}$ pin "Low" on the leading edge of E. When the MPU replies by setting BA and BS to a one, that cycle will be a dead cycle used to transfer bus mastership to the DMA controller.

False memory accesses may be prevented during and dead cycles by developing a system $\overline{DMAVMA}$ signal which is "Low" in any cycle when BA has changed.

When BA goes "Low" (either as a result of $\overline{DMA/BREQ}$ = "High" or MPU self-refresh), the DMA device should be taken off the bus. Another dead cycle will elapse before the MPU accesses memory, to allow transfer of bus mastership without contention.

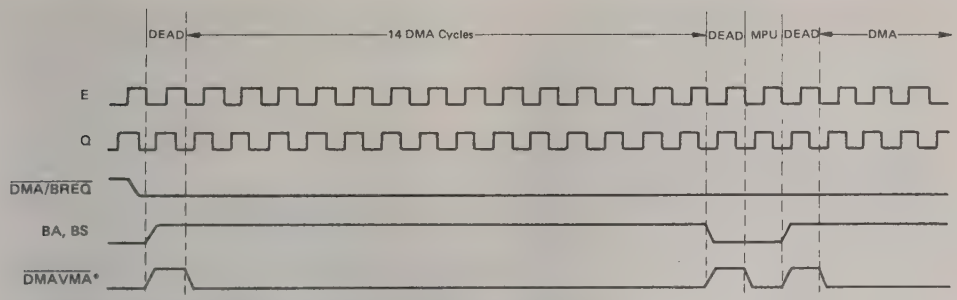
■ MPU OPERATION

During normal operation, the MPU fetches an instruction from memory and then executes the requested function. This



*$\overline{DMAVMA}$ is a signal which is developed externally, but is a system requirement for DMA.

Figure 15 Typical DMA Timing (<14 Cycles)



*DMAVMA is a signal which is developed externally, but is a system requirement for DMA.

Figure 16 Auto – Refresh DMA Timing
(Reverse Cycle Stealing)

sequence begins at RES and is repeated indefinitely unless altered by a special instruction or hardware occurrence. Software instructions that alter normal MPU operation are: SWI, SWI2, SWI3, CWAI, RTI and SYNC. An interrupt, HALT or DMA/BREQ can also alter the normal execution of instructions. Fig. 17 illustrates the flow chart for the HD6809.

■ ADDRESSING MODES

The basic instructions of any computer are greatly enhanced by the presence of powerful addressing modes. The HD6809 has the most complete set of addressing modes available on any microcomputer today. For example, the HD6809 has 59 basic instructions; however, it recognizes 1464 different variations of instructions and addressing modes. The addressing modes support modern programming techniques. The following addressing modes are available on the HD6809:

(1) Implied (Includes Accumulator)
(2) Immediate
(3) Extended
(4) Extended Indirect
(5) Direct
(6) Register
(7) Indexed
    - Zero-Offset
    - Constant Offset
    - Accumulator Offset
    - Auto Increment/Decrement
(8) Indexed Indirect
(9) Relative
(10) Program Counter Relative

• Implied (Includes Accumulator)

In this addressing mode, the opcode of the instruction contains all the address information necessary. Examples of Implied Addressing are: ABX, DAA, SWI, ASRA, and CLRB.

• Immediate Addressing

In Immediate Addressing, the effective address of the data is the location immediately following the opcode (i.e., the data to be used in the instruction immediately follows the opcode of the instruction). The HD6809 uses both 8 and 16-bit immediate values depending on the size of argument specified by the opcode. Examples of instructions with Immediate Addressing are:

```
LDA #$20
LDX #$F000
LDY #CAT
```

(NOTE) # signifies Immediate addressing, $ signifies hexadecimal value.

• Extended Addressing

In Extended Addressing, the contents of the two bytes immediately following the opcode fully specify the 16-bit effective address used by the instruction. Note that the address generated by an extended instruction defines an absolute address and is not position independent. Examples of Extended Addressing include:

```
LDA   CAT
STX   MOUSE
LDD   $2000
```

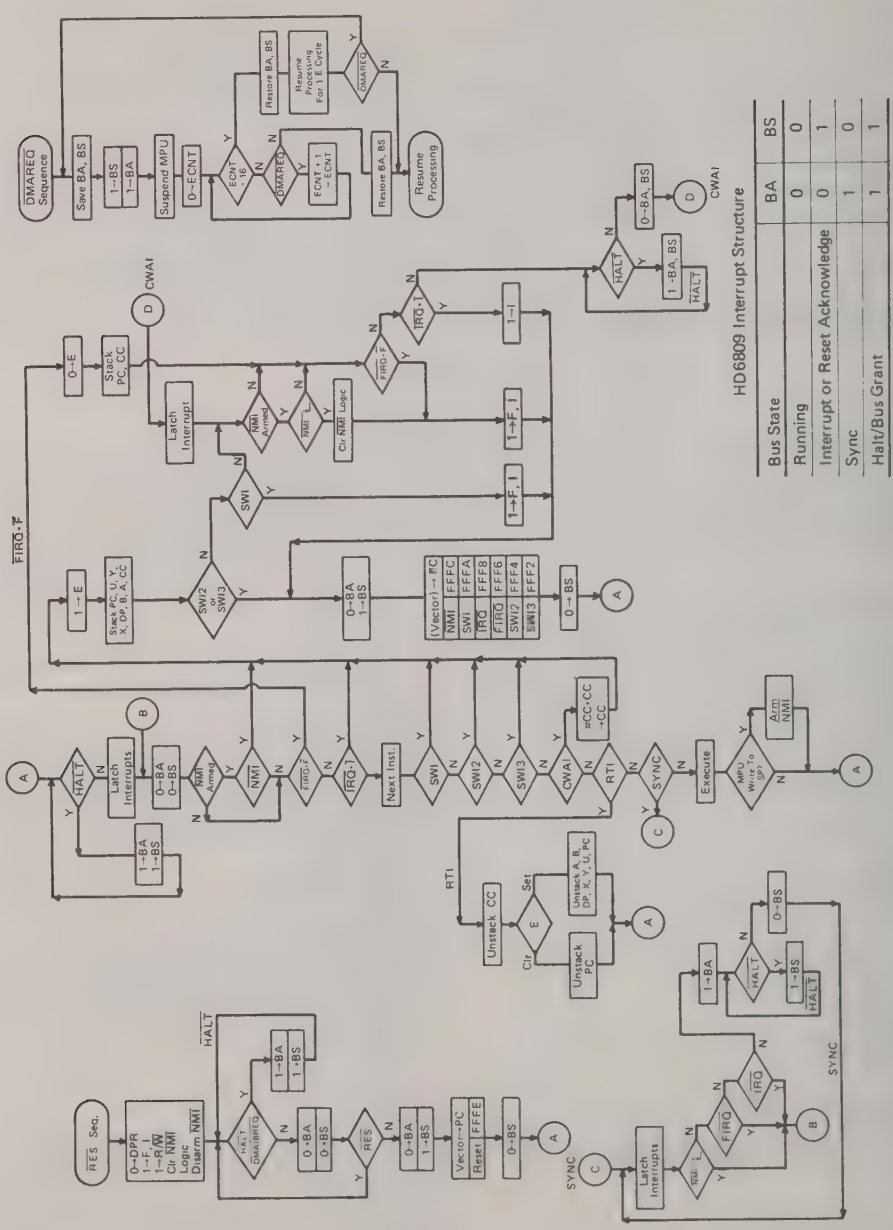
• Extended Indirect

As a special case of indexed addressing (discussed below), "1" level of indirection may be added to Extended Addressing. In Extended Indirect, the two bytes following the postbyte of an Indexed instruction contain the address of the data.

```
LDA   [CAT]
LDX   [$FFFE]
STU   [DOG]
```

• Direct Addressing

Direct addressing is similar to extended addressing except that only one byte of address follows the opcode. This byte specifies the lower 8-bit of the address to be used. The upper 8-bit of the address are supplied by the direct page register. Since only one byte of address is required in direct addressing, this mode requires less memory and executes faster than extended addressing. Of course, only 256 locations (one page) can be



HD6809 Interrupt Structure

Bus State	BA	BS
Running	0	0
Interrupt or Reset Acknowledge	0	1
Sync	1	0
Halt/Bus Grant	1	1

(NOTE) Asserting $\overline{RES}$ will result in entering the reset sequence from any point in the flow chart.

Figure 17 Flowchart for HD6809 Instruction

accessed without redefining the contents of the DP register. Since the DP register is set to $00 on Reset, direct addressing on the HD6809 is compatible with direct addressing on the HD6800. Indirection is not allowed in direct addressing. Some examples of direct addressing are:

```
LDA     $30
SETDP   $10 (Assembler directive)
LDB     $1030
LDD     <CAT
```

(NOTE) < is an assembler directive which forces direct addressing.

- **Register Addressing**

Some opcodes are followed by a byte that defines a register or set of registers to be used by the instruction. This is called a postbyte. Some examples of register addressing are:

```
TFR     X, Y        Transfers X into Y
EXG     A, B        Exchanges A with B
PSHS    A, B, X, Y  Push Y, X, B and A onto S
PULU    X, Y, D     Pull D, X, and Y from U
```

- **Indexed Addressing**

In all indexed addressing, one of the pointer registers (X, Y, U, S, and sometimes PC) is used in a calculation of the effective address of the operand to be used by the instruction. Five basic types of indexing are available and are discussed below. The postbyte of an indexed instruction specifies the basic type and variation of the addressing mode as well as the pointer register to be used. Fig. 18 lists the legal formats for the postbyte. Table 3 gives the assembler form and the number of cycles and bytes

Post-Byte Register Bit								Indexed Addressing Mode
7	6	5	4	3	2	1	0	
0	R	R	×	×	×	×	×	EA = ,R + 5 Bit Offset
1	R	R	0	0	0	0	0	,R +
1	R	R	0/1	0	0	0	1	,R + +
1	R	R	0	0	0	1	0	, -R
1	R	R	0/1	0	0	1	1	, - - R
1	R	R	0/1	0	1	0	0	EA = ,R + 0 Offset
1	R	R	0/1	0	1	0	1	EA = ,R + ACCB Offset
1	R	R	0/1	0	1	1	0	EA = ,R + ACCA Offset
1	R	R	0/1	1	0	0	0	EA = , R + 8 Bit Offset
1	R	R	0/1	1	0	0	1	EA = ,R + 16 Bit Offset
1	R	R	0/1	1	0	1	1	EA = ,R + D Offset
1	×	×	0/1	1	1	0	0	EA = ,PC + 8 Bit Offset
1	×	×	0/1	1	1	0	1	EA = ,PC + 16 Bit Offset
1	R	R	1	1	1	1	1	EA = [,Address]

Addressing Mode Field

Indirect Field
(Sigh bit when b7 = 0)
{0 ..... Non Indirect
{1 ..... Indirect

Register Field : RR
00 = X
01 = Y
10 = U
11 = S

× = Don't Care

Figure 18 Index Addressing Postbyte Register Bit Assignments

Table 3 Indexed Addressing Mode

Type	Forms	Non Indirect				Indirect			
		Assembler Form	Postbyte OP Code	+ ~	+ #	Assembler Form	Postbyte OP Code	+ ~	+ #
Constant Offset From R (2's Complement Offsets)	No Offset	,R	1RR00100	0	0	[,R]	1RR10100	3	0
	5 Bit Offset	n, R	0RRnnnnn	1	0	defaults to 8-bit			
	8 Bit Offset	n, R	1RR01000	1	1	[n, R]	1RR11000	4	1
	16 Bit Offset	n, R	1RR01001	4	2	[n, R]	1RR11001	7	2
Accumulator Offset From R (2's Complement Offsets)	A Register Offset	A, R	1RR00110	1	0	[A, R]	1RR10110	4	0
	B Register Offset	B, R	1RR00101	1	0	[B, R]	1RR10101	4	0
	D Register Offset	D, R	1RR01011	4	0	[D, R]	1RR11011	7	0
Auto Increment/Decrement R	Increment By 1	,R +	1RR00000	2	0	not allowed			
	Increment By 2	,R + +	1RR00001	3	0	[,R + +]	1RR10001	6	0
	Decrement By 1	, - R	1RR00010	2	0	not allowed			
	Decrement By 2	, - - R	1RR00011	3	0	[, - - R]	1RR10011	6	0
Constant Offset From PC (2's Complement Offsets)	8 Bit Offset	n, PCR	1××01100	1	1	[n, PCR]	1××11100	4	1
	16 Bit Offset	n, PCR	1××01101	5	2	[n, PCR]	1××11101	8	2
Extended Indirect	16 Bit Address	–	–	–	–	[n]	10011111	5	2

R = X, Y, U or S
× = Don't Care

RR:
00 = X
01 = Y
10 = U
11 = S

$\overset{+}{\sim}$ and $\overset{+}{\#}$ indicate the number of additional cycles and bytes for the particular variation.

added to the basic values for indexed addressing for each variation.

Zero-Offset Indexed

In this mode, the selected pointer register contains the effective address of the data to be used by the instruction. This is the fastest indexing mode.

Examples are:

```
LDD   0,X
LDA   S
```

Constant Offset Indexed

In this mode, a two's-complement offset and the contents of one of the pointer registers are added to form the effective address of the operand. The pointer register's initial content is unchanged by the addition.

Three sizes of offsets are available:

5-bit (−16 to +15)
8-bit (−128 to +127)
16-bit (−32768 to +32767)

The two's complement 5-bit offset is included in the postbyte and, therefore, is most efficient in use of bytes and cycles. The two's complement 8-bit offset is contained in a single byte following the postbyte. The two's complement 16-bit offset is in the two bytes following the postbyte. In most cases the programmer need not be concerned with the size of this offset since the assembler will select the optimal size automatically.

Examples of constant-offset indexing are:

```
LDA   23,X
LDX   −2,S
LDY   300,X
LDU   CAT,Y
```

Accumulator-Offset Indexed

This mode is similar to constant offset indexed except that the two's-complement value in one of the accumulators (A, B or D) and the contents of one of the pointer registers are added to form the effective address of the operand. The contents of both the accumulator and the pointer register are unchanged by the addition. The postbyte specifies which accumulator to use as an offset and no additional bytes are required. The advantage of an accumulator offset is that the value of the offset can be calculated by a program at run-time.

Some examples are:

```
LDA   B,Y
LDX   D,Y
LEAX  B,X
```

Auto Increment/Decrement Indexed

In the auto increment addressing mode, the pointer register contains the address of the operand. Then, after the pointer register is used it is incremented by one or two. This addressing mode is useful in stepping through tables, moving data, or for the creation of software stacks. In auto decrement, the pointer register is decremented prior to use as the address of the data. The use of auto decrement is similar to that of auto increment; but the tables, etc., are scanned from the "High" to "Low" addresses. The size of the increment/decrement can be either one or two to allow for tables of either 8 or 16-bit data to be accessed and is selectable by the programmer. The pre-decrement, post-increment nature of these modes allow them to be used to create additional software stacks that behave identically to the U and S stacks.

Some examples of the auto increment/decrement addressing modes are:

```
LDA   ,X+
STD   ,Y++
LDB   ,−Y
LDX   ,−−S
```

Care should be taken in performing operations on 16-bit pointer registers (X, Y, U, S) where the same register is used to calculate the effective address.

Consider the following instruction:

STX 0, X++ (X initialized to 0)

The desired result is to store a 0 in locations $0000 and $0001 then increment X to point to $0002. In reality, the following occurs:

0 → temp	calculate the EA; temp is a holding register
X + 2 → X	perform autoincrement
X → (temp)	do store operation

• Indexed Indirect

All of the indexing modes with the exception of auto increment/decrement by one, or a ±4-bit offset may have an additional level of indirection specified. In indirect addressing, the effective address is contained at the location specified by the contents of the Index register plus any offset. In the example below, the A accumulator is loaded indirectly using an effective address calculated from the Index register and an offset.

```
        Before Execution
        A = ×× (don't care)
        X = $F000
$0100   LDA [$10,X]      EA is now $F010

$F010   $F1              $F150 is now the
$F011   $50              new EA

$F150   $AA
        After Execution
        A = $AA Actual Data Loaded
        X = $F000
```

All modes of indexed indirect are included except those which are meaningless (e.g., auto increment/decrement by 1 indirect). Some examples of indexed indirect are:

```
LDA   [,X]
LDD   [10,S]
LDA   [B,Y]
LDD   [,X++]
```

• Relative Addressing

The byte(s) following the branch opcode is (are) treated as a signed offset which may be added to the program counter. If the branch condition is true then the calculated address (PC + signed offset) is loaded into the program counter. Program execution continues at the new location as indicated by the PC; short (1 byte offset) and long (2 bytes offset) relative addressing modes are available. All of memory can be reached in long relative addressing as an effective address is interpreted modulo 2^{16}. Some examples of relative addressing are:

```
        BEQ     CAT     (short)
        BGT     DOG     (short)
CAT     LBEQ    RAT     (long)
DOG     LBGT    RABBIT  (long)
```

```
        .
        .
        .
RAT     NOP
RABBIT  NOP
```

• Program Counter Relative

The PC can be used as the pointer register with 8 or 16-bit signed offsets. As in relative addressing, the offset is added to the current PC to create the effective address. The effective address is then used as the address of the operand or data. Program Counter Relative Addressing is used for writing position independent programs. Tables related to a particular routine will maintain the same relationship after the routine is moved, if referenced relative to the Program Counter. Examples are:

```
LDA   CAT, PCR
LEAX  TABLE, PCR
```

Since program counter relative is a type of indexing, an additional level of indirection is available.

```
LDA   [CAT, PCR]
LDU   [DOG, PCR]
```

■ HD6809 INSTRUCTION SET

The instruction set of the HD6809 is similar to that of the HD6800 and is upward compatible at the source code level. The number of opcodes has been reduced from 72 to 59, but because of the expanded architecture and additional addressing modes, the number of available opcodes (with different addressing modes) has risen from 197 to 1464.

Some of the new instructions and addressing modes are described in detail below:

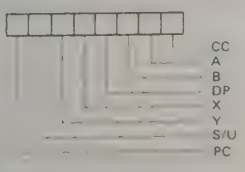
• PSHU/PSHS

The push instructions have the capability of pushing onto either the hardware stack (S) or user stack (U) any single register, or set of registers with a single instruction.

• PULU/PULS

The pull instructions have the same capability of the push instruction, in reverse order. The byte immediately following the push or pull opcode determines which register or registers are to be pushed or pulled. The actual PUSH/PULL sequence is fixed; each bit defines a unique register to push or pull, as shown in below.

PUSH/PULL POST BYTE




```
        ← Pull Order          Push Order →
PC    U   Y   X   DP  B   A   CC
FFFF...← increasing memory address .....0000
PC    S   Y   X   DP  B   A   CC
```

•TFR/EXG

Within the HD6809, any register may be transferred to or exchanged with another of like-size; i.e., 8-bit to 8-bit or 16-bit to 16-bit. Bits 4-7 of postbyte define the source register, while bits 0-3 represent the destination register. Three are denoted as follows:

```
0000 – D        0101 – PC
0001 – X        1000 – A
0010 – Y        1001 – B
0011 – U        1010 – CC
0100 – S        1011 – DP
```

(NOTE) All other combinations are undefined and INVALID.

TRANSFER/EXCHANGE POST BYTE

• LEAX/LEAY/LEAU/LEAS

The LEA (Load Effective Address) works by calculating the effective address used in an indexed instruction and stores that address value, rather than the data at that address, in a pointer register. This makes all the features of the internal addressing hardware available to the programmer. Some of the implications of this instruction are illustrated in Table 4.

The LEA instruction also allows the user to access data in a position independent manner. For example:

```
        LEAX    MSG1, PCR
        LBSR    PDATA (Print message routine)
          .
          .
MSG1 FCC        'MESSAGE'
```

This sample program prints: 'MESSAGE'. By writing MSG1, PCR, the assembler computes the distance between the present address and MSG1. This result is placed as a constant into the LEAX instruction which will be indexed from the PC value at the time of execution. No matter where the code is located, when it is executed, the computed offset from the PC will put the absolute address of MSG1 into the X pointer register. This code is totally position independent.

The LEA instructions are very powerful and use an internal holding register (temp). Care must be exercised when using the LEA instructions with the autoincrement and autodecrement addressing modes due to the sequence of internal operations. The LEA internal sequence is outlined as follows:

```
LEAa, b+        (any of the 16-bit pointer registers X, Y, U
                 or S may be substituted for a and b.)
1. b → temp     (calculate the EA)
2. b + 1 → b    (modify b, postincrement)
3. temp → a     (load a)

LEAa, – b
1. b – 1 → temp (calculate EA with predecrement)
2. b – 1 → b    (modify b, predecrement)
3. temp → a     (load a)
```

Autoincrement-by-two and autodecrement-by-two instructions work similarly. Note that LEAX, X+ does not change X, however LEAX, –X does decrement X. LEAX 1, X should be used to increment X by one.

Table 4 LEA Examples

Instruction	Operation	Comment
LEAX 10, X	X + 10 → X	Adds 5-bit constant 10 to X
LEAX 500, X	X + 500 → X	Adds 16-bit constant 500 to X
LEAY A, Y	Y + A → Y	Adds 8-bit accumulator to Y
LEAY D, Y	Y + D → Y	Adds 16-bit D accumulator to Y
LEAU −10, U	U − 10 → U	Subtracts 10 from U
LEAS −10, S	S − 10 → S	Used to reserve area on stack
LEAS 10, S	S + 10 → S	Used to 'clean up' stack
LEAX 5, S	S + 5 → X	Transfers as well as adds

- **MUL**

Multiplies the unsigned binary numbers in the A and B accumulator and places the unsigned result into the 16-bit D accumulator. This unsigned multiply also allows multiple-precision multiplications.

Long And Short Relative Branches

The HD6809 has the capability of program counter relative branching throughout the entire memory map. In this mode, if the branch is to be taken, the 8 or 16-bit signed offset is added to the value of the program counter to be used as the effective address. This allows the program to branch anywhere in the 64k memory map. Position independent code can be easily generated through the use of relative branching. Both short (8-bit) and long (16-bit) branches are available.

- **SYNC**

After encountering a Sync instruction, the MPU enters a Sync state, stops processing instructions and waits for an interrupt. If the pending interrupt is non-maskable ($\overline{NMI}$) or maskable ($\overline{FIRQ}$, $\overline{IRQ}$) with its mask bit (F or I) clear, the processor will clear the Syne state and perform the normal interrupt stacking and service routine. Since $\overline{FIRQ}$ and $\overline{IRQ}$ are not edge-triggered, a "Low" level with a minimum duration of three bus cycles is required to assure that the interrupt will be taken. If the pending interrupt is maskable ($\overline{FIRQ}$, $\overline{IRQ}$) with its mask bit (F or I) set, the processor will clear the Sync state and continue processing by executing the next inline instruction. Fig. 19 depicts Sync timing.

Software Interrupts

A Software Interrupt is an instruction which will cause an interrupt, and its associated vector fetch. These Software Interrupts are useful in operating system calls, software debugging, trace operations, memory mapping, and software development systems. Three levels of SWI are available on this HD6809, and are prioritized in the following order: SWI, SWI2, SWI3.

16-Bit Operation

The HD6809 has the capability of processing 16-bit data. These instructions include loads, stores, compares, adds, subtracts, transfers, exchanges, pushes and pulls.

■ CYCLE-BY-CYCLE OPERATION

The address bus cycle-by-cycle performance chart illustrates the memory-access sequence corresponding to each possible instruction and addressing mode in the HD6809. Each instruction begins with an opcode fetch. While that opcode is being internally decoded, the next program byte is always fetched. (Most instructions will use the next byte, so this technique considerably speeds throughput.) Next, the operation of each opcode will follow the flow chart. $\overline{VMA}$ is an indication of $FFFF_{16}$ on the address bus, R/$\overline{W}$="High" and BS="Low". The following examples illustrate the use of the chart; see Fig. 20.

Example 1: LBSR (Branch Taken)

Before Execution SP = F000

```
                .
                .
                .
$8000           LBSR    CAT
                .
                .
                .
$A000   CAT     .
```

CYCLE-BY-CYCLE FLOW

Cycle #	Address	Data	R/$\overline{W}$	Description
1	8000	17	1	Opcode Fetch
2	8001	1F	1	Offset High Byte
3	8002	FD	1	Offset Low Byte
4	FFFF	*	1	$\overline{VMA}$ Cycle
5	FFFF	*	1	$\overline{VMA}$ Cycle
6	A000	*	1	Computed Branch Address
7	FFFF	*	1	$\overline{VMA}$ Cycle
8	EFFF	03	0	Stack Low Order Byte of Return Address
9	EFFE	80	0	Stack High Order Byte of Return Address

Example 2: DEC (Extended)

```
$8000   DEC     $A000
$A000   FCB     $80
```

CYCLE-BY-CYCLE FLOW

Cycle #	Address	Data	R/$\overline{W}$	Description
1	8000	7A	1	Opcode Fetch
2	8001	A0	1	Operand Address, High Byte
3	8002	00	1	Operand Address, Low Byte
4	FFFF	*	1	$\overline{VMA}$ Cycle
5	A000	80	1	Read the Data
6	FFFF	*	1	$\overline{VMA}$ Cycle
7	A000	7F	0	Store the Decremented Data

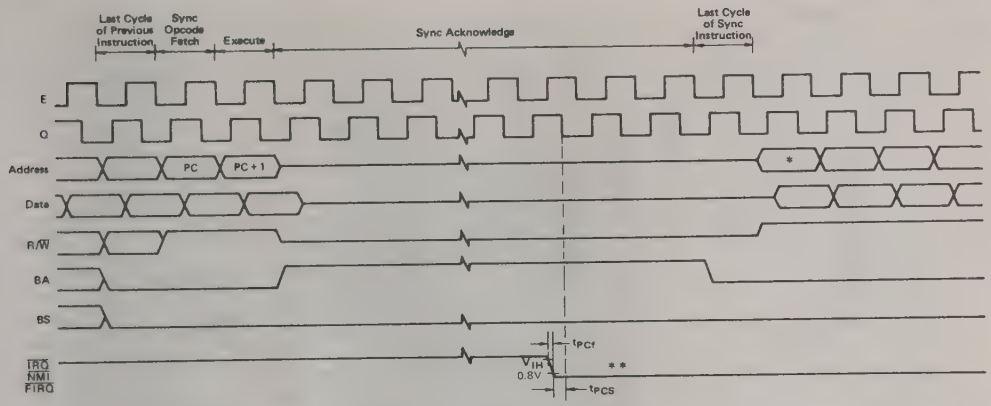
* The data bus has the data at that particular address.

■ HD6809 INSTRUCTION SET TABLES

The instructions of the HD6809 have been broken down into five different categories. They are as follows:

8-Bit operation (Table 5)
16-Bit operation (Table 6)
Index register/stack pointer instructions (Table 7)
Relative branches (long or short) (Table 8)
Miscellaneous instructions (Table 9)

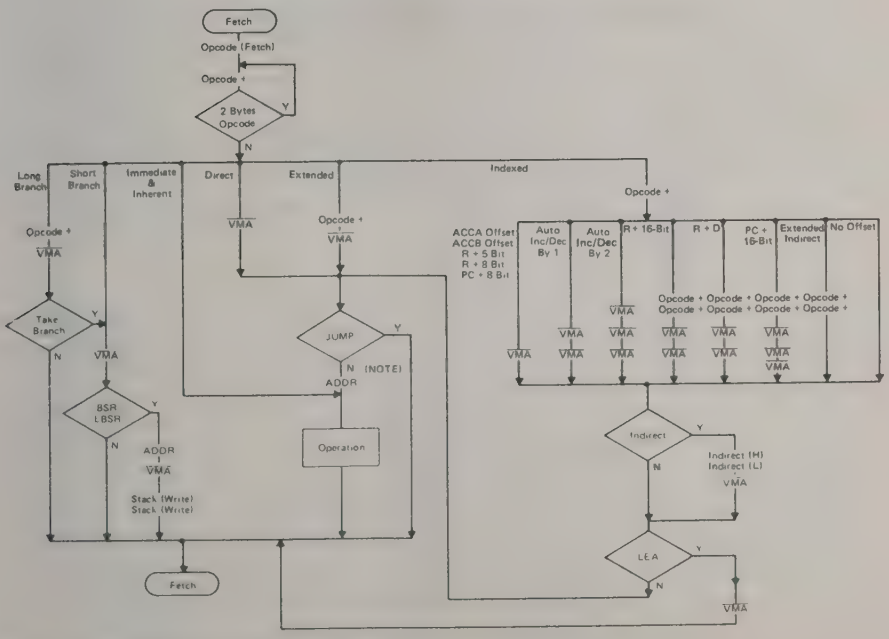
HD6809 instruction set tables and Hexadecimal Values of instructions are shown in Table 10 and Table 11.



(NOTES) • If the associated mask bit is set when the interrupt is requested, this cycle will be an instruction fetch from address location PC + 1. However, if the interrupt is accepted (NMI or an unmasked FIRQ or IRQ) interrupt processing continues with this cycle as (m) on Figure 9 and 10 (Interrupt Timing).

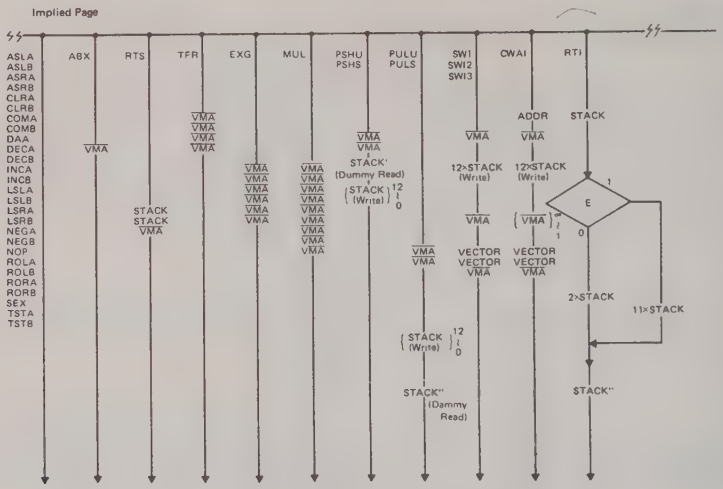
•• If mask bits are clear, IRQ and FIRQ must be held "Low" for three cycles to guarantee that interrupt will be taken, although only one cycle is necessary to bring the processor out of SYNC.

Figure 19 Sync Timing



(NOTE) Write operation during store instruction.

Figure 20 Address Bus Cycle-by-Cycle Performance



(NOTE) STACK' : Address stored in stack pointer before execution.
STACK'': Address set to stack pointer as the result of the execution.

Figure 20 Address Bus Cycle-by-Cycle Performance (Continued)

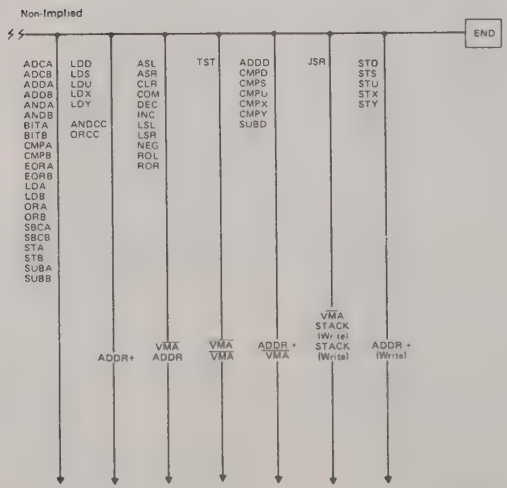


Figure 20 Address Bus Cycle-by-Cycle Performance (Continued)

Table 5 8-Bit Accumulator and Memory Instructions

Mnemonic(s)	Operation
ADCA, ADCB	Add memory to accumulator with carry
ADDA, ADDB	Add memory to accumulator
ANDA, ANDB	And memory with accumulator
ASL, ASLA, ASLB	Arithmetic shift of accumulator or memory left
ASR, ASRA, ASRB	Arithmetic shift of accumulator or memory right
BITA, BITB	Bit test memory with accumulator
CLR, CLRA, CLRB	Clear accumulator or memory location
CMPA, CMPB	Compare memory from accumulator
COM, COMA, COMB	Complement accumultor or memory location
DAA	Decimal adjust A accumulator
DEC, DECA, DECB	Decrement accumulator or memory location
EORA, EORB	Exclusive or memory with accumulator
EXG R1, R2	Exchange R1 with R2 (R1, R2 = A, B, CC, DP)
INC, INCA, INCB	Increment accumulator or memory location
LDA, LDB	Load accumulator from memory
LSL, LSLA, LSLB	Logical shift left accumulator or memory location
LSR, LSRA, LSRB	Logical shift right accumulator or memory location
MUL	Unsigned multiply (A × B → D)
NEG, NEGA, NEGB	Negate accumulator or memory
ORA, ORB	Or memory with accumulator
ROL, ROLA, ROLB	Rotate accumulator or memory left
ROR, RORA, RORB	Rotate accumulator or memory right
SBCA, SBCB	Subtract memory from accumulator with borrow
STA, STB	Store accumulator to memory
SUBA, SUBB	Subtract memory from accumulator
TST, TSTA, TSTB	Test accumulator or memory location
TFR R1, R2	Transfer R1 to R2 (R1, R2 = A, B, CC, DP)

(NOTE) A, B, CC or DP may be pushed to (pulled from) either stack with PSHS, PSHU (PULS, PULU) instructions.

Table 6 16-Bit Accumulator and Memory Instructions

Mnemonic(s)	Operation
ADDD	Add memory to D accumulator
CMPD	Compare memory from D accumulator
EXG D, R	Exchange D with X, Y, S, U or PC
LDD	Load D accumulator from memory
SEX	Sign Extend B accumulator into A accumulator
STD	Store D accumulator to memory
SUBD	Subtract memory from D accumulator
TFR D, R	Transfer D to X, Y, S, U or PC
TFR R, D	Transfer X, Y, S, U or PC to D

(NOTE) D may be pushed (pulled) to either stack with PSHS, PSHU (PULS, PULU) instructions.

Table 7 Index Register/Stack Pointer Instructions

Mnemonic(s)	Operation
CMPS, CMPU	Compare memory from stack pointer
CMPX, CMPY	Compare memory from index register
EXG R1, R2	Exchange D, X, Y, S, U or PC with D, X, Y, S, U or PC
LEAS, LEAU	Load effective address into stack pointer
LEAX, LEAY	Load effective address into index register
LDS, LDU	Load stack pointer from memory
LDX, LDY	Load index register from memory
PSHS	Push A, B, CC, DP, D, X, Y, U, or PC onto hardware stack
PSHU	Push A, B, CC, DP, D, X, Y, S, or PC onto user stack
PULS	Pull A, B, CC, DP, D, X, Y, U or PC from hardware stack
PULU	Pull A, B, CC, DP, D, X, Y, S or PC from user stack
STS, STU	Store stack pointer to memory
STX, STY	Store index register to memory
TFR R1, R2	Transfer D, X, Y, S, U or PC to D, X, Y, S, U or PC
ABX	Add B accumulator to X (unsigned)

Table 8 Branch Instructions

Mnemonic(s)	Operation
	SIMPLE BRANCHES
BEQ, LBEQ	Branch if equal
BNE, LBNE	Branch if not equal
BMI, LBMI	Branch if minus
BPL, LBPL	Branch if plus
BCS, LBCS	Branch if carry set
BCC, LBCC	Branch if carry clear
BVS, LBVS	Branch if overflow set
BVC, LBVC	Branch if overflow clear
	SIGNED BRANCHES
BGT, LBGT	Branch if greater (signed)
BGE, LBGE	Branch if greater than or equal (signed)
BEQ, LBEQ	Branch if equal
BLE, LBLE	Branch if less than or equal (signed)
BLT, LBLT	Branch if less than (signed)
	UNSIGNED BRANCHES
BHI, LBHI	Branch if higher (unsigned)
BHS, LBHS	Branch if higher or same (unsigned)
BEQ, LBEQ	Branch if equal
BLS, LBLS	Branch if lower or same (unsigned)
BLO, LBLO	Branch if lower (unsigned)
	OTHER BRANCHES
BSR, LBSR	Branch to subroutine
BRA, LBRA	Branch always
BRN, LBRN	Branch never

Table 9 Miscellaneous Instructions

Mnemonic(s)	Operation
ANDCC	AND condition code register
CWAI	AND condition code register, then wait for interrupt
NOP	No operation
ORCC	OR condition code register
JMP	Jump
JSR	Jump to subroutine
RTI	Return from interrupt
RTS	Return from subroutine
SWI, SWI2, SWI3	Software interrupt (absolute indirect)
SYNC	Synchronize with interrupt line

Table 10 HD6809 Instruction Set Table

INSTRUCTION/ FORMS		HD6809 ADDRESSING MODES: IMPLIED OP	IMPLIED ~	IMPLIED #	DIRECT OP	DIRECT ~	DIRECT #	EXTENDED OP	EXTENDED ~	EXTENDED #	IMMEDIATE OP	IMMEDIATE ~	IMMEDIATE #	INDEXED① OP	INDEXED ~	INDEXED #	RELATIVE OP	RELATIVE ~②	RELATIVE #	DESCRIPTION	5 H	3 N	2 Z	1 V	0 C
ABX		3A	3	1																B + X → X (UNSIGNED)	•	•	•	•	•
ADC	ADCA				99	4	2	B9	5	3	89	2	2	A9	4+	2+				A + M + C → A	↕	↕	↕	↕	↕
	ADCB				D9	4	2	F9	5	3	C9	2	2	E9	4+	2+				B + M + C → B	↕	↕	↕	↕	↕
ADD	ADDA				9B	4	2	BB	5	3	8B	2	2	AB	4+	2+				A + M → A	↕	↕	↕	↕	↕
	ADDB				DB	4	2	FB	5	3	CB	2	2	EB	4+	2+				B + M → B	↕	↕	↕	↕	↕
	ADDD				D3	6	2	F3	7	3	C3	4	3	E3	6+	2+				D + M:M + 1 → D	•	↕	↕	↕	↕
AND	ANDA				94	4	2	B4	5	3	84	2	2	A4	4+	2+				A ∧ M → A	•	↕	↕	0	•
	ANDB				D4	4	2	F4	5	3	C4	2	2	E4	4+	2+				B ∧ M → B	•	↕	↕	0	•
	ANDCC										1C	3	2							CC ∧ IMM → CC	(			↕	)
ASL	ASLA	48	2	1																A	⑧	↕	↕	↕	↕
	ASLB	58	2	1																B	⑧	↕	↕	↕	↕
	ASL				08	6	2	78	7	3				68	6+	2+				M	⑧	↕	↕	↕	↕
ASR	ASRA	47	2	1																A	⑧	↕	↕	•	↕
	ASRB	57	2	1																B	⑧	↕	↕	•	↕
	ASR				07	6	2	77	7	3				67	6+	2+				M	⑧	↕	↕	•	↕
BCC	BCC																24	3	2	Branch C = 0	•	•	•	•	•
	LBCC																10 24	5(6)	4	Long Branch C = 0	•	•	•	•	•
BCS	BCS																25	3	2	Branch C = 1	•	•	•	•	•
	LBCS																10 25	5(6)	4	Long Branch C = 1	•	•	•	•	•
BEQ	BEQ																27	3	2	Branch Z = 1	•	•	•	•	•
	LBEQ																10 27	5(6)	4	Long Branch Z = 1	•	•	•	•	•
BGE	BGE																2C	3	2	Branch N ⊕ V=0	•	•	•	•	•
	LBGE																10 2C	5(6)	4	Long Branch N ⊕ V=0	•	•	•	•	•
BGT	BGT																2E	3	2	Branch Z∨(N⊕V)=0	•	•	•	•	•
	LBGT																10 2E	5(6)	4	Long Branch Z∨(N ⊕ V)=0	•	•	•	•	•
BHI	BHI																22	3	2	Branch C∨Z=0	•	•	•	•	•
	LBHI																10 22	5(6)	4	Long Branch C∨Z=0	•	•	•	•	•
BHS	BHS																24	3	2	Branch C=0	•	•	•	•	•
	LBHS																10 24	5(6)	4	Long Branch C=0	•	•	•	•	•
BIT	BITA				95	4	2	B5	5	3	85	2	2	A5	4+	2+				Bit Test A (M ∧ A)	•	↕	↕	0	•
	BITB				D5	4	2	F5	5	3	C5	2	2	E5	4+	2+				Bit Test B (M ∧ B)	•	↕	↕	0	•
BLE	BLE																2F	3	2	Branch Z∨(N⊕V)=1	•	•	•	•	•
	LBLE																10 2F	5(6)	4	Long Branch Z∨(N ⊕ V)=1	•	•	•	•	•
BLO	BLO																25	3	2	Branch C=1	•	•	•	•	•
	LBLO																10 25	5(6)	4	Long Branch C=1	•	•	•	•	•
BLS	BLS																23	3	2	Branch C∨Z=1	•	•	•	•	•
	LBLS																10 23	5(6)	4	Long Branch C∨Z=1	•	•	•	•	•
BLT	BLT																2D	3	2	Branch N ⊕ V=1	•	•	•	•	•
	LBLT																10 2D	5(6)	4	Long Branch N ⊕ V=1	•	•	•	•	•
BMI	BMI																2B	3	2	Branch N=1	•	•	•	•	•
	LBMI																10 2B	5(6)	4	Long Branch N=1	•	•	•	•	•
BNE	BNE																26	3	2	Branch Z = 0	•	•	•	•	•
	LBNE																10 26	5(6)	4	Long Branch Z = 0	•	•	•	•	•
BPL	BPL																2A	3	2	Branch N = 0	•	•	•	•	•
	LBPL																10 2A	5(6)	4	Long Branch N = 0	•	•	•	•	•
BRA	BRA																20	3	2	Branch Always	•	•	•	•	•
	LBRA																16	5	3	Long Branch/ Always	•	•	•	•	•
BRN	BRN																21	3	2	Branch Never	•	•	•	•	•
	LBRN																10 21	5	4	Long Branch Never	•	•	•	•	•

(to be continued)

INSTRUCTION/ FORMS		HD6809 ADDRESSING MODES																	DESCRIPTION	5	3	2	1	0
		IMPLIED			DIRECT			EXTENDED			IMMEDIATE			INDEXED①			RELATIVE				H	N	Z	V	C
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~③	#						
BSR	BSR																8D	7	2	Branch to Subroutine	•	•	•	•	•
	LBSR																17	9	3	Long Branch to Subroutine	•	•	•	•	•
BVC	BVC																28	3	2	Branch V = 0	•	•	•	•	•
	LBVC																10 28	5(6)	4	Long Branch V = 0	•	•	•	•	•
BVS	BVS																29	3	2	Branch V = 1	•	•	•	•	•
	LBVS																10 29	5(6)	4	Long Branch V = 1	•	•	•	•	•
CLR	CLRA	4F	2	1																0 → A	•	0	1	0	0
	CLRB	5F	2	1																0 → B	•	0	1	0	0
	CLR				0F	6	2	7F	7	3				6F	6+	2+				0 → M	•	0	1	0	0
CMP	CMPA				91	4	2	B1	5	3	81	2	2	A1	4+	2+				Compare M from A	⑧	↕	↕	↕	↕
	CMPB				D1	4	2	F1	5	3	C1	2	2	E1	4+	2+				Compare M from B	⑧	↕	↕	↕	↕
	CMPD				10 93	7	3	10 B3	8	4	10 83	5	4	10 A3	7+	3+				Compare M: M + 1 from D	•	↕	↕	↕	↕
	CMPS				11 9C	7	3	11 BC	8	4	11 8C	5	4	11 AC	7+	3+				Compare M: M + 1 from S	•	↕	↕	↕	↕
	CMPU				11 93	7	3	11 B3	8	4	11 83	5	4	11 A3	7+	3+				Compare M: M + 1 from U	•	↕	↕	↕	↕
	CMPX				9C	6	2	BC	7	3	8C	4	3	AC	6+	2+				Compare M: M + 1 from X	•	↕	↕	↕	↕
	CMPY				10 9C	7	3	10 BC	8	4	10 8C	5	4	10 AC	7+	3+				Compare M: M + 1 from Y	•	↕	↕	↕	↕
COM	COMA	43	2	1																$\overline{A}$ → A	•	↕	↕	0	1
	COMB	53	2	1																$\overline{B}$ → B	•	↕	↕	0	1
	COM				03	6	2	73	7	3				63	6+	2+				$\overline{M}$ → M	•	↕	↕	0	1
CWAI		3C	20	2																CC ∧ IMM → CC (except 1→E) Wait for Interrupt	(——⑦——)				
DAA		19	2	1																Decimal Adjust A	•	↕	↕	⑧	↕
DEC	DECA	4A	2	1																A − 1 → A	•	↕	↕	↕	•
	DECB	5A	2	1																B − 1 → B	•	↕	↕	↕	•
	DEC				0A	6	2	7A	7	3				6A	6+	2+				M − 1 → M	•	↕	↕	↕	•
EOR	EORA				98	4	2	B8	5	3	88	2	2	A8	4+	2+				A ⊕ M → A	•	↕	↕	0	•
	EORB				D8	4	2	F8	5	3	C8	2	2	E8	4+	2+				B ⊕ M → B	•	↕	↕	0	•
EXG	R1, R2	1E	7	2																R1 ↔ R2②	(——⑩——)				
INC	INCA	4C	2	1																A + 1 → A	•	↕	↕	↕	•
	INCB	5C	2	1																B + 1 → B	•	↕	↕	↕	•
	INC				0C	6	2	7C	7	3				6C	6+	2+				M + 1 → M	•	↕	↕	↕	•
JMP					0E	3	2	7E	4	3				6E	3+	2+				EA③ → PC	•	•	•	•	•
JSR					9D	7	2	BD	8	3				AD	7+	2+				Jump to Subroutine	•	•	•	•	•
LD	LDA				96	4	2	B6	5	3	86	2	2	A6	4+	2+				M → A	•	↕	↕	0	•
	LDB				D6	4	2	F6	5	3	C6	2	2	E6	4+	2+				M → B	•	↕	↕	0	•
	LDD				DC	5	2	FC	6	3	CC	3	3	EC	5+	2+				M: M + 1 → D	•	↕	↕	0	•
	LDS				10 DE	6	3	10 FE	7	4	10 CE	4	4	10 EE	6+	3+				M: M + 1 → S	•	↕	↕	0	•
	LDU				DE	5	2	FE	6	3	CE	3	3	EE	5+	2+				M: M + 1 → U	•	↕	↕	0	•
	LDX				9E	5	2	BE	6	3	8E	3	3	AE	5+	2+				M: M + 1 → X	•	↕	↕	0	•
	LDY				10 9E	6	3	10 BE	7	4	10 8E	4	4	10 AE	6+	3+				M: M + 1 → Y	•	↕	↕	0	•
LEA	LEAS													32	4+	2+				EA③ → S	•	•	•	•	•
	LEAU													33	4+	2+				EA③ → U	•	•	•	•	•
	LEAX													30	4+	2+				EA③ → X	•	•	↕	•	•
	LEAY													31	4+	2+				EA③ → Y	•	•	↕	•	•
LSL	LSLA	48	2	1																A } B } M }	•	↕	↕	↕	↕
	LSLB	58	2	1																	•	↕	↕	↕	↕
	LSL				08	6	2	78	7	3				68	6+	2+					•	↕	↕	↕	↕
LSR	LSRA	44	2	1																A } B } M }	•	0	↕	•	↕
	LSRB	54	2	1																	•	0	↕	•	↕
	LSR				04	6	2	74	7	3				64	6+	2+					•	0	↕	•	↕
MUL		3D	11	1																A × B → D (Unsigned)	•	•	↕	•	⑨
NEG	NEGA	40	2	1																$\overline{A}$ + 1 → A	⑧	↕	↕	↕	↕
	NEGB	50	2	1																$\overline{B}$ + 1 → B	⑧	↕	↕	↕	↕
	NEG				00	6	2	70	7	3				60	6+	2+				$\overline{M}$ + 1 → M	⑧	↕	↕	↕	↕
NOP		12	2	1																No Operation	•	•	•	•	•

(to be continued)

INSTRUCTION/ FORMS		HD6809 ADDRESSING MODES																	DESCRIPTION	5	3	2	1	0
		IMPLIED			DIRECT			EXTENDED			IMMEDIATE			INDEXED①			RELATIVE				H	N	Z	V	C
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~⑤	#						
OR	ORA				9A	4	2	BA	5	3	8A	2	2	AA	4+	2+				A ∨ M → A	•	↕	↕	0	•
	ORB				DA	4	2	FA	5	3	CA	2	2	EA	4+	2+				B ∨ M → B	•	↕	↕	0	•
	ORCC										1A	3	2							CC ∨ IMM → CC	(		⑦		)
PSH	PSHS	34	5+④	2																Push Registers on S Stack	•	•	•	•	•
	PSHU	36	5+④	2																Push Registers on U Stack	•	•	•	•	•
PUL	PULS	35	5+④	2																Pull Registers from S Stack	(		⑩		)
	PULU	37	5+④	2																Pull Registers from U Stack	(		⑩		)
ROL	ROLA	49	2	1																A	•	↕	↕	↕	↕
	ROLB	59	2	1																B	•	↕	↕	↕	↕
	ROL				09	6	2	79	7	3				69	6+	2+				M	•	↕	↕	↕	↕
ROR	RORA	46	2	1																A	•	↕	↕	•	↕
	RORB	56	2	1																B	•	↕	↕	•	↕
	ROR				06	6	2	76	7	3				66	6+	2+				M	•	↕	↕	•	↕
RTI		3B	6/15	1																Return From Interrupt	(		⑦		)
RTS		39	5	1																Return From Subroutine	•	•	•	•	•
SBC	SBCA				92	4	2	B2	5	3	82	2	2	A2	4+	2+				A − M − C → A	⑧	↕	↕	↕	↕
	SBCB				D2	4	2	F2	5	3	C2	2	2	E2	4+	2+				B − M − C → B	⑧	↕	↕	↕	↕
SEX		1D	2	1																Sign Extend B into A	•	↕	↕	•	•
ST	STA				97	4	2	B7	5	3				A7	4+	2+				A → M	•	↕	↕	0	•
	STB				D7	4	2	F7	5	3				E7	4+	2+				B → M	•	↕	↕	0	•
	STD				DD	5	2	FD	6	3				ED	5+	2+				D → M: M+1	•	↕	↕	0	•
	STS				10 DF	6	3	10 FF	7	4				10 EF	6+	3+				S → M: M+1	•	↕	↕	0	•
	STU				DF	5	2	FF	6	3				EF	5+	2+				U → M: M+1	•	↕	↕	0	•
	STX				9F	5	2	BF	6	3				AF	5+	2+				X → M: M+1	•	↕	↕	0	•
	STY				10 9F	6	3	10 BF	7	4				10 AF	6+	3+				Y → M: M+1	•	↕	↕	0	•
SUB	SUBA				90	4	2	B0	5	3	80	2	2	A0	4+	2+				A − M → A	⑧	↕	↕	↕	↕
	SUBB				D0	4	2	F0	5	3	C0	2	2	E0	4+	2+				B − M → B	⑧	↕	↕	↕	↕
	SUBD				93	6	2	B3	7	3	83	4	3	A3	6+	2+				D − M: M+1 → D	•	↕	↕	↕	↕
SWI	SWI⑥	3F	19	1																Software Interrupt1	•	•	•	•	•
	SWI2⑥	10 3F	20	2																Software Interrupt2	•	•	•	•	•
	SWI3⑥	11 3F	20	2																Software Interrupt3	•	•	•	•	•
SYNC		13	≥2	1																Synchronize to Interrupt	•	•	•	•	•
TFR	R1, R2	1F	6	2																R1 → R2②	(		⑩		)
TST	TSTA	4D	2	1																Test A	•	↕	↕	0	•
	TSTB	5D	2	1																Test B	•	↕	↕	0	•
	TST				0D	6	2	7D	7	3				6D	6+	2+				Test M	•	↕	↕	0	•

(NOTES)

① This column gives a base cycle and byte count. To obtain total count, and the values obtained from the INDEXED ADDRESSING MODES table.
② R1 and R2 may be any pair of 8 bit or any pair of 16 bit registers.
The 8 bit registers are: A, B, CC, DP
The 16 bit registers are: X, Y, U, S, D, PC
③ EA is the effective address.
④ The PSH and PUL instructions require 5 cycle plus 1 cycle for each byte pushed or pulled.
⑤ 5(6) means: 5 cycles if branch not taken, 6 cycles if taken.
⑥ SWI sets I and F bits. SWI2 and SWI3 do not affect I and F.
⑦ Conditions Codes set as a direct result of the instruction.
⑧ Value of half-carry flag is undefined.
⑨ Special Case — Carry set if b7 is SET.
⑩ Condition Codes set as a direct result of the instruction if CC is specified, and not affected otherwise.

LEGEND:

OP	Operation Code (Hexadecimal)	Z	Zero (byte)
~	Number of MPU Cycles	V	Overflow, 2's complement
#	Number of Program Bytes	C	Carry from bit 7
+	Arithmetic Plus	↕	Test and set if true, cleared otherwise
−	Arithmetic Minus	•	Not Affected
×	Multiply	CC	Condition Code Register
M̄	Complement of M	:	Concatenation
→	Transfer Into	∨	Logical or
H	Half-carry (from bit 3)	∧	Logical and
N	Negative (sign bit)	⊕	Logical Exclusive or

Table 11 Hexadecimal Values of Machine Codes

OP	Mnem	Mode	~	#	OP	Mnem	Mode	~	#	OP	Mnem	Mode	~	#
00	NEG	Direct	6	2	30	LEAX	Indexed	4+	2+	60	NEG	Indexed	6+	2+
01	*				31	LEAY		4+	2+	61	*			
02	*				32	LEAS		4+	2+	62	*			
03	COM		6	2	33	LEAU	Indexed	4+	2+	63	COM		6+	2+
04	LSR		6	2	34	PSHS	Implied	5+	2	64	LSR		6+	2+
05	*				35	PULS		5+	2	65	*			
06	ROR		6	2	36	PSHU		5+	2	66	ROR		6+	2+
07	ASR		6	2	37	PULU		5+	2	67	ASR		6+	2+
08	ASL, LSL		6	2	38	*				68	ASL, LSL		6+	2+
09	ROL		6	2	39	RTS		5	1	69	ROL		6+	2+
0A	DEC		6	2	3A	ABX		3	1	6A	DEC		6+	2+
0B	*				3B	RTI		6, 15	1	6B	*			
0C	INC		6	2	3C	CWAI		20	2	6C	INC		6+	2+
0D	TST		6	2	3D	MUL		11	1	6D	TST		6+	2+
0E	JMP		3	2	3E	*				6E	JMP		3+	2+
0F	CLR	Direct	6	2	3F	SWI	Implied	19	1	6F	CLR	Indexed	6+	2+
10	See	—	—	—	40	NEGA	Implied	2	1	70	NEG	Extended	7	3
11	Next Page	—	—	—	41	*				71	*			
12	NOP	Implied	2	1	42	*				72	*			
13	SYNC	Implied	2	1	43	COMA		2	1	73	COM		7	3
14	*				44	LSRA		2	1	74	LSR		7	3
15	*				45	*				75	*			
16	LBRA	Relative	5	3	46	RORA		2	1	76	ROR		7	3
17	LBSR	Relative	9	3	47	ASRA		2	1	77	ASR		7	3
18	*				48	ASLA, LSLA		2	1	78	ASL, LSL		7	3
19	DAA	Implied	2	1	49	ROLA		2	1	79	ROL		7	3
1A	ORCC	Immed	3	2	4A	DECA		2	1	7A	DEC		7	3
1B	*				4B	*				7B	*			
1C	ANDCC	Immed	3	2	4C	INCA		2	1	7C	INC		7	3
1D	SEX	Implied	2	1	4D	TSTA		2	1	7D	TST		7	3
1E	EXG		8	2	4E	*				7E	JMP		4	3
1F	TFR	Implied	6	2	4F	CLRA	Implied	2	1	7F	CLR	Extended	7	3
20	BRA	Relative	3	2	50	NEGB	Implied	2	1	80	SUBA	Immed	2	2
21	BRN		3	2	51	*				81	CMPA		2	2
22	BHI		3	2	52	*				82	SBCA		2	2
23	BLS		3	2	53	COMB		2	1	83	SUBD		4	3
24	BHS, BCC		3	2	54	LSRB		2	1	84	ANDA		2	2
25	BLO, BCS		3	2	55	*				85	BITA		2	2
26	BNE		3	2	56	RORB		2	1	86	LDA		2	2
27	BEQ		3	2	57	ASRB		2	1	87	*			
28	BVC		3	2	58	ASLB, LSLB		2	1	88	EORA		2	2
29	BVS		3	2	59	ROLB		2	1	89	ADCA		2	2
2A	BPL		3	2	5A	DECB		2	1	8A	ORA		2	2
2B	BMI		3	2	5B	*				8B	ADDA		2	2
2C	BGE		3	2	5C	INCB		2	1	8C	CMPX	Immed	4	3
2D	BLT		3	2	5D	TSTB		2	1	8D	BSR	Relative	7	2
2E	BGT		3	2	5E	*				8E	LDX	Immed	3	3
2F	BLE	Relative	3	2	5F	CLRB	Implied	2	1	8F	*			

LEGEND:
~ Number of MPU cycles (less possible push pull or indexed-mode cycles)
# Number of program bytes
* Denotes unused opcode

(to be continued)

OP	Mnem	Mode	~	#
90	SUBA	Direct	4	2
91	CMPA		4	2
92	SBCA		4	2
93	SUBD		6	2
94	ANDA		4	2
95	BITA		4	2
96	LDA		4	2
97	STA		4	2
98	EORA		4	2
99	ADCA		4	2
9A	ORA		4	2
9B	ADDA		4	2
9C	CMPX		6	2
9D	JSR		7	2
9E	LDX		5	2
9F	STX	Direct	5	2
A0	SUBA	Indexed	4+	2+
A1	CMPA		4+	2+
A2	SBCA		4+	2+
A3	SUBD		6+	2+
A4	ANDA		4+	2+
A5	BITA		4+	2+
A6	LDA		4+	2+
A7	STA		4+	2+
A8	EORA		4+	2+
A9	ADCA		4+	2+
AA	ORA		4+	2+
AB	ADDA		4+	2+
AC	CMPX		6+	2+
AD	JSR		7+	2+
AE	LDX		5+	2+
AF	STX	Indexed	5+	2+
B0	SUBA	Extended	5	3
B1	CMPA		5	3
B2	SBCA		5	3
B3	SUBD		7	3
B4	ANDA		5	3
B5	BITA		5	3
B6	LDA		5	3
B7	STA		5	3
B8	EORA		5	3
B9	ADCA		5	3
BA	ORA		5	3
BB	ADDA		5	3
BC	CMPX		7	3
BD	JSR		8	3
BE	LDX		6	3
BF	STX	Extended	6	3
C0	SUBB	Immed	2	2
C1	CMPB		2	2
C2	SBCB		2	2
C3	ADDD		4	3
C4	ANDB		2	2
C5	BITB	Immed	2	2

OP	Mnem	Mode	~	#
C6	LDB	Immed	2	2
C7	*			
C8	EORB		2	2
C9	ADCB		2	2
CA	ORB		2	2
CB	ADDB		2	2
CC	LDD		3	3
CD	*			
CE	LDU	Immed	3	3
CF	*			
D0	SUBB	Direct	4	2
D1	CMPB		4	2
D2	SBCB		4	2
D3	ADDD		6	2
D4	ANDB		4	2
D5	BITB		4	2
D6	LDB		4	2
D7	STB		4	2
D8	EORB		4	2
D9	ADCB		4	2
DA	ORB		4	2
DB	ADDB		4	2
DC	LDD		5	2
DD	STD		5	2
DE	LDU		5	2
DF	STU	Direct	5	2
E0	SUBB	Indexed	4+	2+
E1	CMPB		4+	2+
E2	SBCB		4+	2+
E3	ADDD		6+	2+
E4	ANDB		4+	2+
E5	BITB		4+	2+
E6	LDB		4+	2+
E7	STB		4+	2+
E8	EORB		4+	2+
E9	ADCB		4+	2+
EA	ORB		4+	2+
EB	ADDB		4+	2+
EC	LDD		5+	2+
ED	STD		5+	2+
EE	LDU		5+	2+
EF	STU	Indexed	5+	2+
F0	SUBB	Extended	5	3
F1	CMPB		5	3
F2	SBCB		5	3
F3	ADDD		7	3
F4	ANDB		5	3
F5	BITB		5	3
F6	LDB		5	3
F7	STB		5	3
F8	EORB		5	3
F9	ADCB		5	3
FA	ORB		5	3
FB	ADDB	Extended	5	3

OP	Mnem	Mode	~	#
FC	LDD	Extended	6	3
FD	STD		6	3
FE	LDU		6	3
FF	STU	Extended	6	3

2 Bytes Opcode

1021	LBRN	Relative	5	4
1022	LBHI		5(6)	4
1023	LBLS		5(6)	4
1024	LBHS, LBCC		5(6)	4
1025	LBCS, LBLO		5(6)	4
1026	LBNE		5(6)	4
1027	LBEQ		5(6)	4
1028	LBVC		5(6)	4
1029	LBVS		5(6)	4
102A	LBPL		5(6)	4
102B	LBMI		5(6)	4
102C	LBGE		5(6)	4
102D	LBLT		5(6)	4
102E	LBGT		5(6)	4
102F	LBLE	Relative	5(6)	4
103F	SWI2	Implied	20	2
1083	CMPD	Immed	5	4
108C	CMPY		5	4
108E	LDY	Immed	4	4
1093	CMPD	Direct	7	3
109C	CMPY		7	3
109E	LDY		6	3
109F	STY	Direct	6	3
10A3	CMPD	Indexed	7+	3+
10AC	CMPY		7+	3+
10AE	LDY		6+	3+
10AF	STY	Indexed	6+	3+
10B3	CMPD	Extended	8	4
10BC	CMPY		8	4
10BE	LDY		7	4
10BF	STY	Extended	7	4
10CE	LDS	Immed	4	4
10DE	LDS	Direct	6	3
10DF	STS	Direct	6	3
10EE	LDS	Indexed	6+	3+
10EF	STS	Indexed	6+	3+
10FE	LDS	Extended	7	4
10FF	STS	Extended	7	4
113F	SWI3	Implied	20	2
1183	CMPU	Immed	5	4
118C	CMPS	Immed	5	4
1193	CMPU	Direct	7	3
119C	CMPS	Direct	7	3
11A3	CMPU	Indexed	7+	3+
11AC	CMPS	Indexed	7+	3+
11B3	CMPU	Extended	8	4
11BC	CMPS	Extended	8	4

(NOTE): All unused opcodes are both undefined and illegal

■ NOTE FOR USE

[1] **Exceptional Operation of HD6809**

(a) **Exceptional Operations of $\overline{\text{DMA/BREQ}}$, BA signals (#1)**

HD6809 acknowledges the input signal level of $\overline{\text{DMA/BREQ}}$ at the end of each cycle, then determines whether the next sequence is MPU or DMA. When "Low" level is detected, HD6809 executes DMA sequence by setting BA, BS to "High" level. However, in the conditions shown below the assertion of BA, BS delays one clock cycle.

< Conditions for the exception >

(1) $\overline{\text{DMA/BREQ}}$: "Low" for 6~13 cycles
(2) $\overline{\text{DMA/BREQ}}$: "High" for 3 cycles

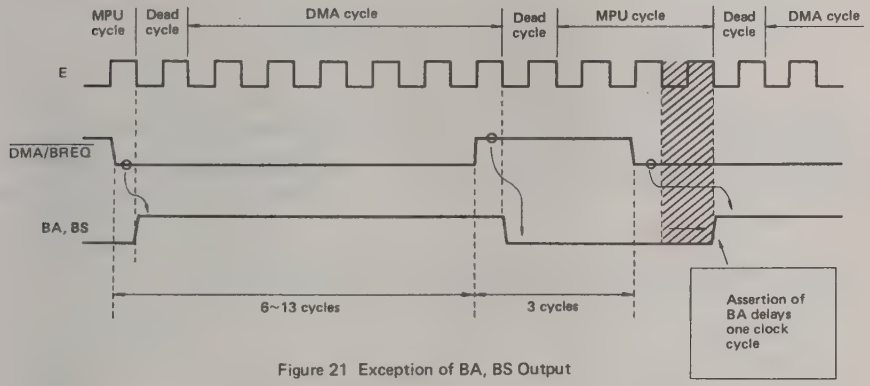


Figure 21 Exception of BA, BS Output

(b) **Exceptional Operations of $\overline{\text{DMA/BREQ}}$, BA signals (#2)**

HD6809 includes a self refresh counter for the reverce cycle steal. And it is only cleared if $\overline{\text{DMA/BREQ}}$ is inactive ("High") for 3 or more MPU cycles. So 1 or 2 inactive cycle(s) doesn't affect the self refresh counter.

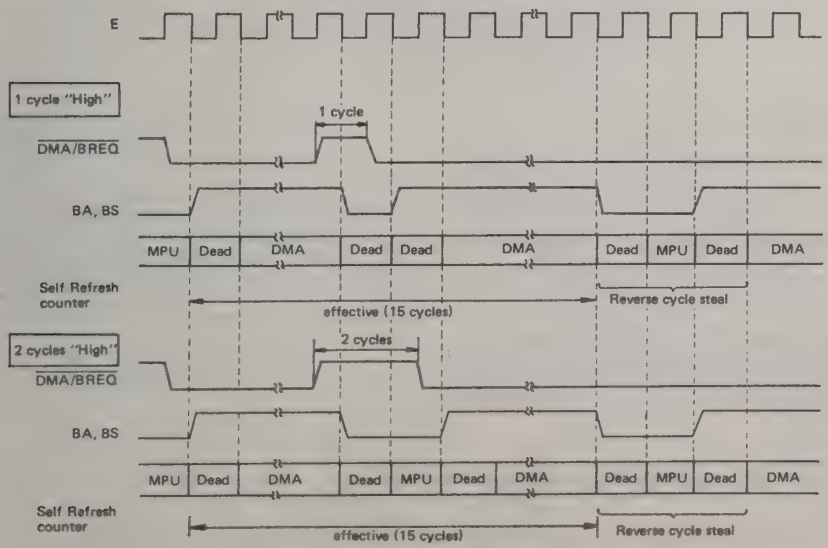


Figure 22 Exception of $\overline{\text{DMA/BREQ}}$

(c) **How to avoid these exceptional operations**

It is necessary to provide 4 or more cycles for inactive $\overline{\text{DMA/BREQ}}$ level as shown in Fig. 23.

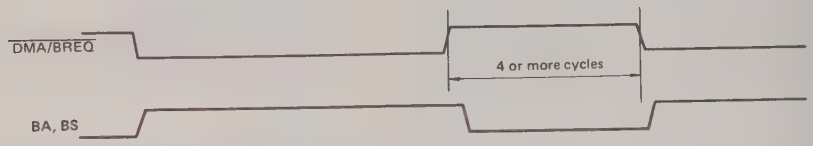


Figure 23 How to Avoid Exceptional Operations

[2] **Restriction for DMA Transfer**

There is a restriction for the DMA transfer in the HD6809 (MPU), HD6844 (DMAC) system. Please take care of following.

(a) **An Example of the System Configuration**

This restriction is applied to the following system.

(1) $\overline{\text{DMA/BREQ}}$ is used for DMA request.

(2) "Halt Burst Mode" is used for DMA transfer

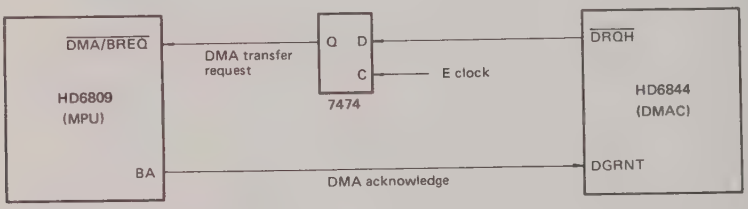


Figure 24 An Example of HD6809, HD6844 System

(The restriction is also applied to the system which doesn't use 7474 Flip-Flop. Fig. 24, Fig. 25 shows an example which uses 7474 for synchronizing DMA request with E.)

(b) **Restriction**

"The number of transfer Byte per one DMA Burst transfer must be less than or equal to 14."

Halt burst DMA transfer should be less than or equal to 14 cycles. In another word, the number stored into DMA Byte count register should be 0~14.

★ Please than care of the section [1](b) if 2 or more DMA channels are used for the DMA transfer.

(c) **Incorrect operation of HD6809, HD6844 system**

"Incorrect Operation" will occur if the number of DMA transfer Byte is more than 14 bytes. If $\overline{\text{DMA/BREQ}}$ is kept in "Low" level HD6809 performs reverse cycle steals once in 14 DMA cycles by taking back the bus control. In this case, however, the action taken by MPU is a little bit different from the DMAC.

As shown in Fig. 25, DMA controller can't stop DMA transfer (Ⓐ) by BA falling edge and excutes an extra DMA cycle during HD6809 dead cycle. So MPU cycle is excuted right after DMA cycle, the Bus confliction occurs at the beginning of MPU cycle.

(d) **How to impliment Halt Bust DMA transfer (> 14 cycles)**

Please use $\overline{\text{HALT}}$ input of HD6809 for the DMA request instead of $\overline{\text{DMA/BREQ}}$.

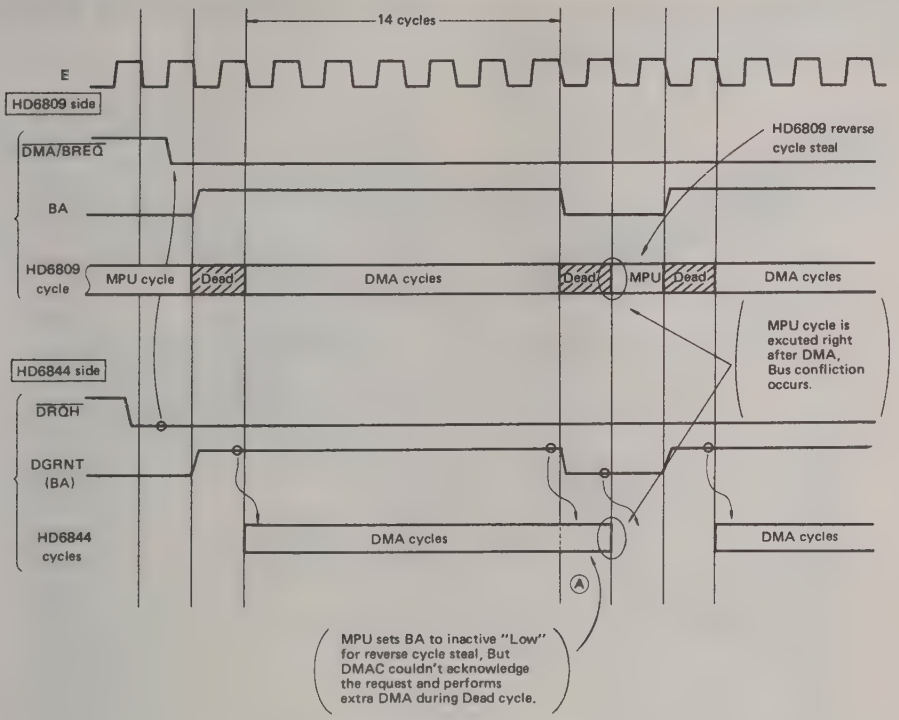


Figure 25 Comparison of HD6809, HD6844 DMA cycles

[3] **Note for CLR Instruction**

Cycle-by-cycle flow of CLR instruction (Direct, Extended, Indexed Addressing Mode) is shown below. In this sequence the content of the memory location specified by the operand is read before writing "00" into it. Note that status Flags, such as IRQ Flag, will be cleared by this extra data read operation when accessing the control/status register (sharing the same address between read and write) of peripheral devices.

Example: CLR (Extended)

$8000	CLR	$A000
$A000	FCB	$80

Cycle #	Address	Data	R/$\overline{W}$	Description
1	8000	7F	1	Opcode Fetch
2	8001	A0	1	Operand Address, High Byte
3	8002	00	1	Operand Address, Low Byte
4	FFFF	*	1	$\overline{VMA}$ Cycle
5	A000	80	1	Read the Data
6	FFFF	*	1	$\overline{VMA}$ Cycle
7	A000	00	0	Store Fixed "00" into Specified Location

* The data bus has the data at that particular address.

[4] **Note for MRDY**

HD6809 require synchronization of the MRDY input with the 4f clock. The synchronization necessitates an external oscillator as shown in Figure 26. The negative transition of the MRDY signal, normally derived from the chip select decoding, must meet the t_{PCS} timing. MRDY's positive transition must occur with the rising edge of 4f.

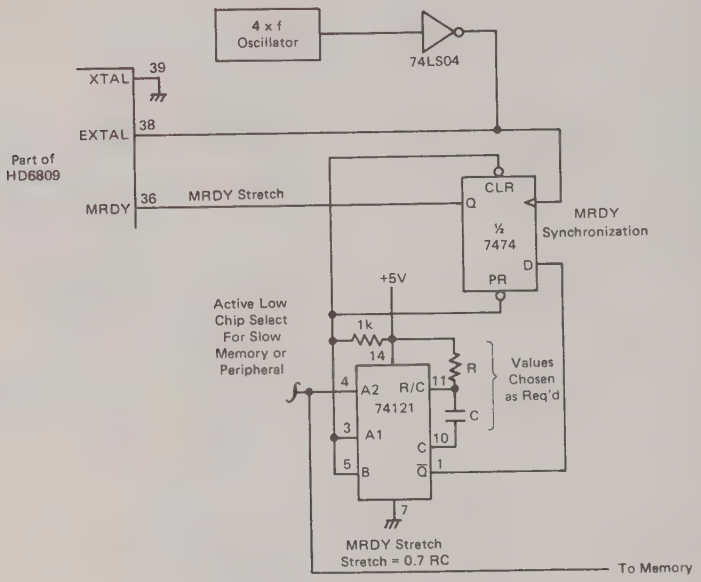


Figure 26 MRDY Synchronization

HD6309
CMOS MPU (Micro Processing Unit)

–ADVANCE INFORMATION–

The HD6309 is the highest 8-bit microprocessor of HMCS6800 family, which is just compatible with the conventional HD6809.

The HD6309 has hardware and software features which make it an ideal processor for higher level language execution or standard controller applications.

The HD6309 is complete CMOS device and the power dissipation is extremely low. Moreover the SYNC and CWAI instruction makes lower power application possible.

HD6309P

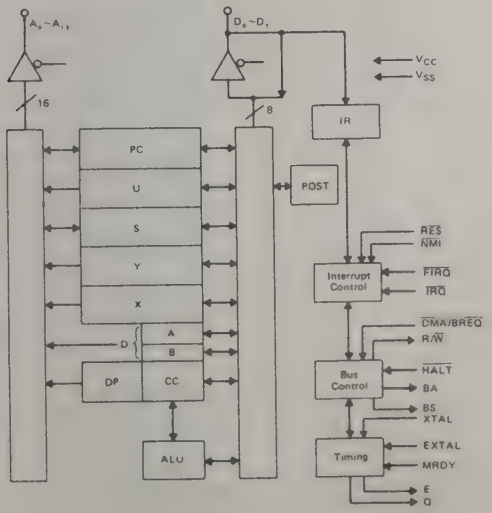
(DP-40)

■ FEATURES

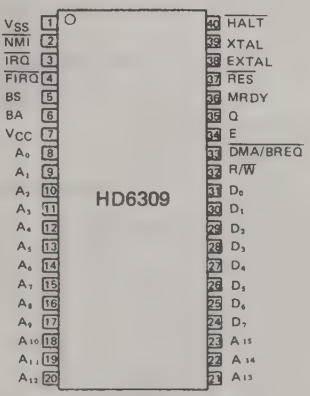
- Hardware – Interfaces with All HMCS6800 Peripherals
  – DMA transfer with no auto-refresh cycle
  – Lin-Limited MRDY control
- Software – Fully Compatible with HD6809, HD6809E, HD6309E MPU families
- Low Power Consumption Mode; SYNC and CWAI instruction
- Crystal Oscillation
- Wide Operation Range
  f= 0.5 to 3 MHz
  (V_{CC} = 5V±10%)

Bus Timing
2.0 MHz
2.5 MHz
3.0 MHz

■ BLOCK DIAGRAM



■ PIN ARRANGEMENT



(Top View)

HD6809E, HD68A09E, HD68B09E

MPU (Micro Processing Unit)

The HD6809E is a revolutionary high performance 8-bit microprocessor which supports modern programming techniques such as position independence, reentrancy, and modular programming.

This third-generation addition to the HMCS6800 family has major architectural improvements which include additional registers, instructions and addressing modes.

The basic instructions of any computer are greatly enhanced by the presence of powerful addressing modes. The HD6809E has the most complete set of addressing modes available on any 8-bit microprocessor today.

The HD6809E has hardware and software features which make it an ideal processor for higher level language execution or standard controller applications. External clock inputs are provided to allow synchronization with peripherals, systems or other MPUs.

HD6800 COMPATIBLE

- Hardware — Interfaces with All HMCS6800 Peripherals
- Software — Upward Source Code Compatible Instruction Set and Addressing Modes

■ **ARCHITECTURAL FEATURES**

- Two 16-bit Index Registers
- Two 16-bit Indexable Stack Pointers
- Two 8-bit Accumulators can be Concatenated to Form One 16-Bit Accumulator
- Direct Page Register Allows Direct Addressing Throughout Memory

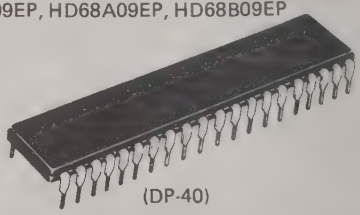
■ **HARDWARE FEATURES**

- External Clock Inputs, E and Q, Allow Synchronization
- TSC Input Controls Internal Bus Buffers
- LIC Indicates Opcode Fetch
- AVMA Allows Efficient Use of Common Resources in A Multiprocessor System
- BUSY is a Status Line for Multiprocessing
- Fast Interrupt Request Input Stacks Only Condition Code Register and Program Counter
- Interrupt Acknowledge Output Allows Vectoring By Devices
- SYNC Acknowledge Output Allows for Synchronization to External Event
- Single Bus-Cycle RESET
- Single 5-Volt Supply Operation
- NMI Blocked After RESET Until After First Load of Stack Pointer
- Early Address Valid Allows Use With Slower Memories
- Early Write-Data for Dynamic Memories

■ **SOFTWARE FEATURES**

- 10 Addressing Modes
  - HMCS6800 Upward Compatible Addressing Modes
  - Direct Addressing Anywhere in Memory Map
  - Long Relative Branches
  - Program Counter Relative
  - True Indirect Addressing
  - Expanded Indexed Addressing:
    0, 5, 8, or 16-bit Constant Offsets
    8, or 16-bit Accumulator Offsets

HD6809EP, HD68A09EP, HD68B09EP

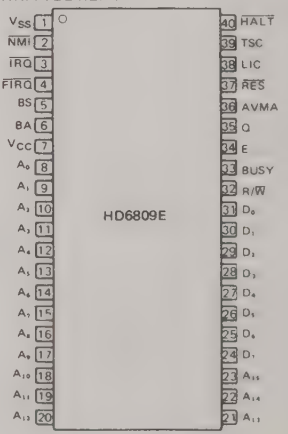


(DP-40)

Auto-Increment/Decrement by 1 or 2

- Improved Stack Manipulation
- 1464 Instruction with Unique Addressing Modes
- 8 x 8 Unsigned Multiply
- 16-bit Arithmetic
- Transfer/Exchange All Registers
- Push/Pull Any Registers or Any Set of Registers
- Load Effective Address

■ **PIN ARRANGEMENT**



(Top View)

■ ABSOLUTE MAXIMUM RATINGS

Item	Symbol	Value	Unit
Supply Voltage	Vcc*	−0.3 ~ +7.0	V
Input Voltage	Vin*	−0.3 ~ +7.0	V
Operating Temperature Range	Topr	−20 ~ +75	°C
Storage Temperature Range	Tstg	−55 ~ +150	°C

* With respect to Vss (SYSTEM GND)

(NOTE) Permanent LSI damage may occur if maximum ratings are exceeded. Normal operation should be under recommended operating conditions. If these conditions are exceeded, it could affect reliability of LSI.

■ RECOMMENDED OPERATING CONDITIONS

Item		Symbol	min	typ	max	unit
Supply Voltage		Vcc*	4.75	5.0	5.25	V
Input Voltage	Logic, Q, $\overline{RES}$	V_{IL}*	−0.2	—	0.8	V
	E	V_{ILC}*	−0.3	—	0.4	V
	Logic	V_{IH}*	2.2	—	Vcc*	V
	$\overline{RES}$		4.0	—	Vcc*	V
	E	V_{IHC}*	Vcc* −0.75	—	Vcc* +0.3	V
Operating Temperature		Topr	−20	25	75	°C

* With respect to Vss (SYSTEM GND)

■ ELECTRICAL CHARACTERISTICS

● DC CHARACTERISTICS (V_{CC} = 5.0V ±5%, V_{SS} = 0V, Ta = −20 ~ +75°C, unless otherwise noted.)

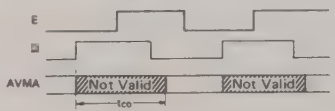
Item		Symbol	Test Condition	HD6809E			HD68A09E			HD68B09E			Unit
				min	typ*	max	min	typ*	max	min	typ*	max	
Input "High" Voltage	Logic, Q	V_{IH}		2.2	—	Vcc	2.2	—	Vcc	2.2	—	Vcc	V
	$\overline{RES}$	V_{IHR}		4.0	—	Vcc	4.0	—	Vcc	4.0	—	Vcc	V
	E	V_{IHC}		Vcc −0.75	—	Vcc +0.3	Vcc −0.75	—	Vcc +0.3	Vcc −0.75	—	Vcc +0.3	V
Input "Low" Voltage	Logic, Q, $\overline{RES}$	V_{IL}		−0.2	—	0.8	−0.2	—	0.8	−0.2	—	0.8	V
	E	V_{ILC}		−0.3	—	0.4	−0.3	—	0.4	−0.3	—	0.4	V
Input Leakage Current	Logic, Q, $\overline{RES}$	I_{in}	Vin = 0 ~ 5.25V, Vcc = max	−2.5	—	2.5	−2.5	—	2.5	−2.5	—	2.5	μA
	E			−100	—	100	−100	—	100	−100	—	100	μA
Output "High" Voltage	$D_0 \sim D_7$	V_{OH}	I_{Load} = −205μA, Vcc = min	2.4	—	—	2.4	—	—	2.4	—	—	V
	$A_0 \sim A_{15}$, R/$\overline{W}$		I_{Load} = −145μA, Vcc = min	2.4	—	—	2.4	—	—	2.4	—	—	V
	BA, BS, LIC, AVMA, BUSY		I_{Load} = −100μA, Vcc = min	2.4	—	—	2.4	—	—	2.4	—	—	V
Output "Low" Voltage		V_{OL}	I_{Load} = 2mA, Vcc = min	—	—	0.5	—	—	0.5	—	—	0.5	V
Power Dissipation		P_D		—	—	1.0	—	—	1.0	—	—	1.0	W
Input Capacitance	$D_0 \sim D_7$, Logic Input, Q, $\overline{RES}$	C_{in}	Vin = 0V, Ta = 25°C, f = 1MHz	—	10	15	—	10	15	—	10	15	pF
	E			—	30	50	—	30	50	—	30	50	pF
Output Capacitance	$A_0 \sim A_{15}$, R/$\overline{W}$, BA, BS, LIC, AVMA, BUSY	C_{out}	Vin = 0V, Ta = 25°C, f = 1MHz	—	10	15	—	10	15	—	10	15	pF
Frequency of Operation	E, Q	f		0.1	—	1.0	0.1	—	1.5	0.1	—	2.0	MHz
Three-State (Off State) Input Current	$D_0 \sim D_7$	I_{TSI}	Vin = 0.4 ~ 2.4V, Vcc = max	−10	—	10	−10	—	10	−10	—	10	μA
	$A_0 \sim A_{15}$, R/$\overline{W}$			−100	—	100	−100	—	100	−100	—	100	μA

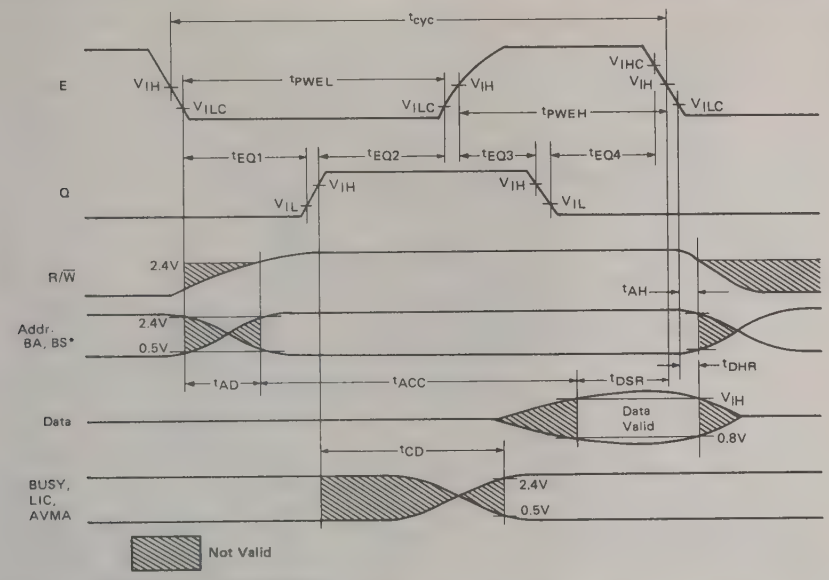
* Ta = 25°C, Vcc = 5V

- **AC CHARACTERISTICS (V_{CC} = 5.0V ±5%, V_{SS} = 0V, Ta = -20 ~ +75°C, unless otherwise noted.)**
  READ/WRITE TIMING

Item		Symbol	Test Condition	HD6809E			HD68A09E			HD68B09E			Unit
				min	typ	max	min	typ	max	min	typ	max	
Cycle Time		t_{cyc}	Fig. 1, 2, 7 ~ 10, 14 and 17	1000	–	10000	667	–	10000	500	–	10000	ns
Peripheral Read Access Times $t_{cyc} - t_{Ef} - t_{AD} - t_{DSR} = t_{ACC}$		t_{ACC}		695	–	–	440	–	–	330	–	–	ns
Data Setup Time (Read)		t_{DSR}		80	–	–	60	–	–	40	–	–	ns
Input Data Hold Time		t_{DHR}		10	–	–	10	–	–	10	–	–	ns
Output Data Hold Time	Ta = 0 ~ +75°C	t_{DHW}		30	–	–	30	–	–	30	–	–	ns
	Ta = -20 ~ 0°C			20	–	–	20	–	–	20	–	–	ns
Address Hold Time (Address, R/$\overline{W}$)	Ta = 0 ~ +75°C	t_{AH}		20	–	–	20	–	–	20	–	–	ns
	Ta = -20 ~ 0°C			10	–	–	10	–	–	10	–	–	ns
Address Delay		t_{AD}		–	–	200	–	–	140	–	–	120	ns
Data Delay Time (Write)		t_{DDW}		–	–	200	–	–	140	–	–	110	ns
E Clock "Low"		t_{PWEL}		450	–	9500	295	–	9500	210	–	9500	ns
E Clock "High" (Measured at V_{IH})		t_{PWEH}		450	–	9500	280	–	9500	220	–	9500	ns
E Rise and Fall Time		t_{Er}, t_{Ef}		–	–	25	–	–	25	–	–	20	ns
Q Clock "High"		t_{PWQH}		450	–	9500	280	–	9500	220	–	9500	ns
Q Rise and Fall Time		t_{Qr}, t_{Qf}		–	–	25	–	–	25	–	–	20	ns
E "Low" to Q Rising		t_{EQ1}		200	–	–	130	–	–	100	–	–	ns
Q "High" to E Rising		t_{EQ2}		200	–	–	130	–	–	100	–	–	ns
E "High" to Q Falling		t_{EQ3}		200	–	–	130	–	–	100	–	–	ns
Q "Low" to E Falling		t_{EQ4}		200	–	–	130	–	–	100	–	–	ns
Interrupts $\overline{HALT}$, $\overline{RES}$ and TSC Setup Time		t_{PCS}		200	–	–	140	–	–	110	–	–	ns
TSC Drive to Valid Logic Levels		t_{TSA}		–	–	210	–	–	150	–	–	120	ns
TSC Release MOS Buffers to High Impedance		t_{TSR}		–	–	200	–	–	140	–	–	110	ns
TSC Three-State Delay		t_{TSD}		–	–	120	–	–	85	–	–	80	ns
Control Delay (BUSY, LIC)		t_{CD}		–	–	300	–	–	250	–	–	200	ns
Control Delay (AVMA*)		t_{CD}		–	–	300	–	–	270	–	–	240	ns
Processor Control Rise/Fall		t_{PCr}, t_{PCf}		–	–	100	–	–	100	–	–	100	ns
TSC Input Delay		t_{PCT}		10	–	–	10	–	–	10	–	–	ns

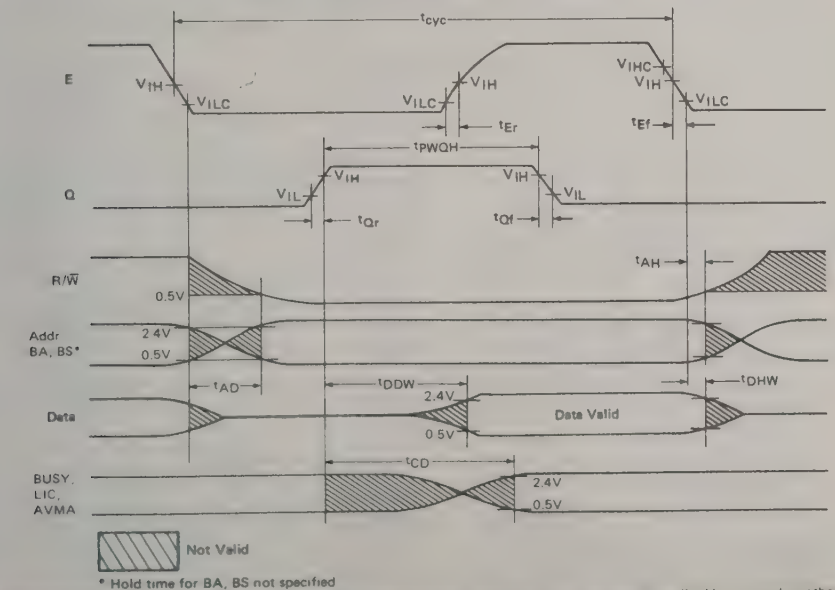
* AVMA drives a not-valid data before providing correct output, so spec t_{CD} max = 270 nsec (HD68A09E) and 240 nsec (HD68B09E) are applied to this signal. When this delay time causes a problem in user's application, please use D-type latch to get stable output.





(NOTE) Waveform measurements for all inputs and outputs are specified at logic "High" = V_{IHmin} and logic "Low" = V_{ILmax} unless otherwise specified.

Figure 1 Read Data from Memory or Peripherals



(NOTE) Waveform measurements for all inputs and outputs are specified at logic "High" = V_{IHmin} and logic "Low" = V_{ILmax} unless otherwise specified.

Figure 2 Write Data to Memory or Peripherals

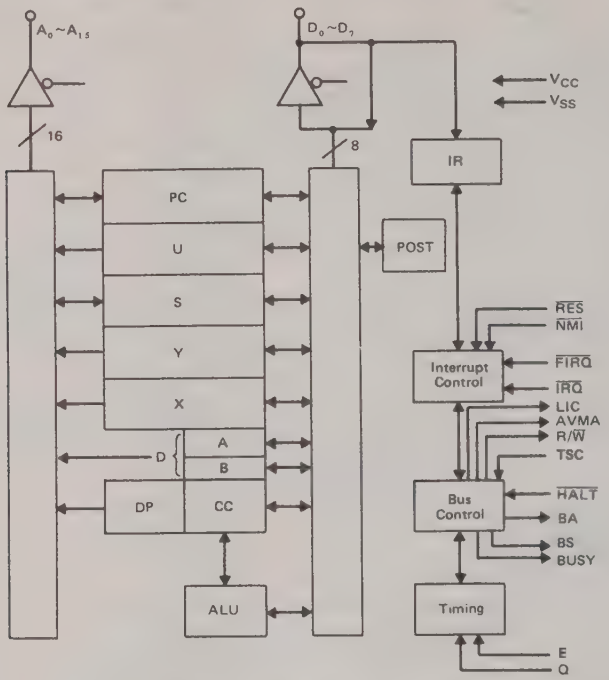
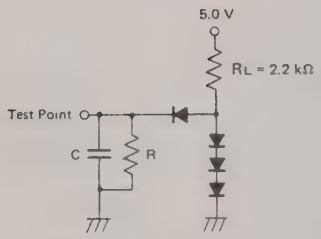


Figure 3 HD6809E Expanded Block Diagram



C = 30 pF for BA, BS, LIC, AVMA, BUSY
130 pF for $D_0 \sim D_7$
90 pF for $A_0 \sim A_{15}$, R/$\overline{W}$

R = 11.7 kΩ for $D_0 \sim D_7$
16.5 kΩ for $A_0 \sim A_{15}$, R/$\overline{W}$
24 kΩ for BA, BS, LIC, AVMA, BUSY

All diodes are 1S2074(H) or equivalent.
C includes stray capacitance.

Figure 4 Bus Timing Test Load

■ PROGRAMMING MODEL

As shown in Figure 5, the HD6809E adds three registers to the set available in the HD6800. The added registers include a Direct Page Register, the User Stack pointer and a second Index Register.

• Accumulators (A, B, D)

The A and B registers are general purpose accumulators which are used for arithmetic calculations and manipulation of data.

Certain instructions concatenate the A and B registers to form a single 16-bit accumulator. This is referred to as the D Register, and is formed with the A Register as the most significant byte.

• Direct Page Register (DP)

The Direct Page Register of the HD6809E serves to enhance the Direct Addressing Mode. The content of this register appears at the higher address outputs ($A_8 \sim A_{15}$) during direct addressing instruction execution. This allows the direct mode to be used at any place in memory, under program control. To ensure HD6800 compatibility, all bits of this register are cleared during Processor Reset.

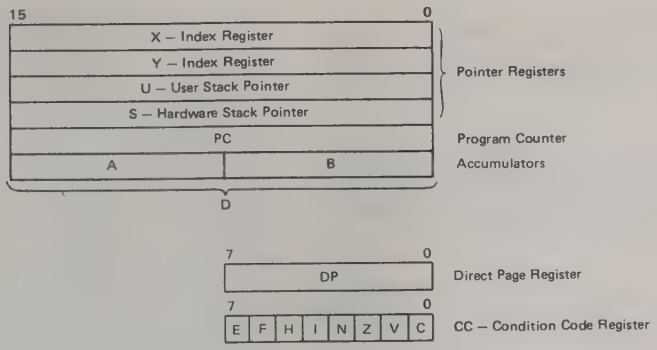


Figure 5 Programming Model of The Microprocessing Unit

- **Index Registers (X, Y)**

The Index Registers are used in indexed mode of addressing. The 16-bit address in this register takes part in the calculation of effective addresses. This address may be used to point to data directly or may be modified by an optional constant or register offset. During some indexed modes, the contents of the index register are incremented or decremented to point to the next item of tabular type data. All four pointer registers (X, Y, U, S) may be used as index registers.

- **Stack Pointer (U, S)**

The Hardware Stack Pointer (S) is used automatically by the processor during subroutine calls and interrupts. The User Stack Pointer (U) is controlled exclusively by the programmer thus allowing arguments to be passed to and from subroutines with ease. The U-register is frequently used as a stack marker. Both Stack Pointers have the same indexed mode addressing capabilities as the X and Y registers, but also support **Push** and **Pull** instructions. This allows the HD6809E to be used efficiently as a stack processor, greatly enhancing its ability to support higher level languages and modular programming.

(NOTE) The stack pointers of the HD6809E point to the top of the stack, in contrast to the HD6800 stack pointer, which pointed to the next free location on stack.

- **Program Counter (PC)**

The Program Counter is used by the processor to point to the address of the next instruction to be executed by the processor. Relative Addressing is provided allowing the Program Counter to be used like an index register in some situations.

- **Condition Code Register (CC)**

The Condition Code Register defines the state of the processor at any given time. See Figure 6.

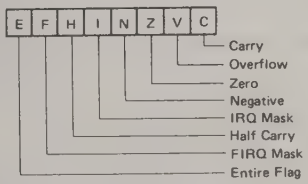


Figure 6 Condition Code Register Format

■ CONDITION CODE REGISTER DESCRIPTION

- **Bit 0 (C)**

Bit 0 is the carry flag, and is usually the carry from the binary ALU. C is also used to represent a 'borrow' from subtract like instructions (CMP, NEG, SUB, SBC) and is the complement of the carry from the binary ALU.

- **Bit 1 (V)**

Bit 1 is the overflow flag, and is set to a one by an operation which causes a signed two's complement arithmetic overflow. This overflow is detected in an operation in which the carry from the MSB in the ALU does not match the carry from the MSB-1.

- **Bit 2 (Z)**

Bit 2 is the zero flag, and is set to a one if the result of the previous operation was identically zero.

- **Bit 3 (N)**

Bit 3 is the negative flag, which contains exactly the value of the MSB of the result of the preceding operation. Thus, a negative two's-complement result will leave N set to a one.

- **Bit 4 (I)**

Bit 4 is the $\overline{IRQ}$ mask bit. The processor will not recognize interrupts from the $\overline{IRQ}$ line if this bit is set to a one. $\overline{NMI}$, $\overline{FIRQ}$, $\overline{IRQ}$, $\overline{RES}$ and SWI all set I to a one; SWI2 and SWI3 do not affect I.

- **Bit 5 (H)**

Bit 5 is the half-carry bit, and is used to indicate a carry from bit 3 in the ALU as a result of an 8-bit addition only (ADC or ADD). This bit is used by the DAA instruction to perform a BCD decimal add adjust operation. The state of this flag is undefined in all subtract-like instructions.

- **Bit 6 (F)**

Bit 6 is the $\overline{FIRQ}$ mask bit. The processor will not recognize interrupts from the $\overline{FIRQ}$ line if this bit is a one. $\overline{NMI}$, $\overline{FIRQ}$, SWI, and $\overline{RES}$ all set F to a one. $\overline{IRQ}$, SWI2 and SWI3 do not affect F.

- **Bit 7 (E)**

Bit 7 is the entire flag, and when set to a one indicates that the complete machine state (all the registers) was stacked, as opposed to the subset state (PC and CC). The E bit of the stacked CC is used on a return from interrupt (RTI) to determine the extent of the unstacking. Therefore, the current E left in the Condition Code Register represents past action.

■ HD6809E MPU SIGNAL DESCRIPTION

- **Power (V_{SS}, V_{CC})**

Two pins are used to supply power to the part: V_{SS} is ground or 0 volts, while V_{CC} is +5.0 V ±5%.

- **Address Bus ($A_0 \sim A_{15}$)**

Sixteen pins are used to output address information from the MPU onto the Address Bus. When the processor does not require the bus for a data transfer, it will output address $FFFF_{16}$, R/$\overline{W}$ = "High", and BS = "Low"; this is a "dummy access" or $\overline{VMA}$ cycle. All address bus drivers are made high-impedance when output Bus Available (BA) is "High" or when TSC is asserted. Each pin will drive one Schottky TTL load or four LS TTL loads, and 90 pF. Refer to Figures 1 and 2.

- **Data Bus ($D_0 \sim D_7$)**

These eight pins provide communication with the system bi-directional data bus. Each pin will drive one Schottky TTL load or four LS TTL loads, and 130 pF.

- **Read/Write (R/$\overline{W}$)**

This signal indicates the direction of data transfer on the data bus. A "Low" indicates that the MPU is writing data onto the data bus. R/$\overline{W}$ is made high impedance when BA is "High" or when TSC is asserted. Refer to Figures 1 and 2.

- **$\overline{RES}$**

A "Low" level on this Schmitt-trigger input for greater than one bus cycle will reset the MPU, as shown in Figure 7. The Reset vectors are fetched from locations $FFFE_{16}$ and $FFFF_{16}$ (Table 1) when Interrupt Acknowledge is true, ($\overline{BA}$ · BS = 1). During initial power-on, the Reset line should be held "Low" until the clock input signals are fully operational.

Because the HD6809E Reset pin has a Schmitt-trigger input with a threshold voltage higher than that of standard peripherals, a simple R/C network may be used to reset the entire system. This higher threshold voltage ensures that all peripherals are out of the reset state before the Processor.

Table 1 Memory Map for Interrupt Vectors

Memory Map for Vector Locations		Interrupt Vector Description
MS	LS	
FFFE	FFFF	$\overline{RES}$
FFFC	FFFD	$\overline{NMI}$
FFFA	FFFB	SWI
FFF8	FFF9	$\overline{IRQ}$
FFF6	FFF7	$\overline{FIRQ}$
FFF4	FFF5	SWI2
FFF2	FFF3	SWI3
FFF0	FFF1	Reserved

- **$\overline{HALT}$**

A "Low" level on this input pin will cause the MPU to stop running at the end of the present instruction and remain halted indefinitely without loss of data. When halted, the BA output is driven "High" indicating the buses are high impedance. BS is also "High" which indicates the processor is in the Halt state. While halted, the MPU will not respond to external real-time requests ($\overline{FIRQ}$, $\overline{IRQ}$) although $\overline{NMI}$ or $\overline{RES}$ will be latched for later response. During the Halt state Q and E should continue to run normally. A halted state ($\overline{BA}$ · BS = 1) can be achieved by pulling $\overline{HALT}$ "Low" while $\overline{RES}$ is still "Low". See Figure 8.

- **Bus Available, Bus Status (BA, BS)**

The Bus Available output is an indication of an internal control signal which makes the MOS buses of the MPU high impedance. When BA goes "Low", a dead cycle will elapse before the MPU acquires the bus. BA will not be asserted when TSC is active, thus allowing dead cycle consistency.

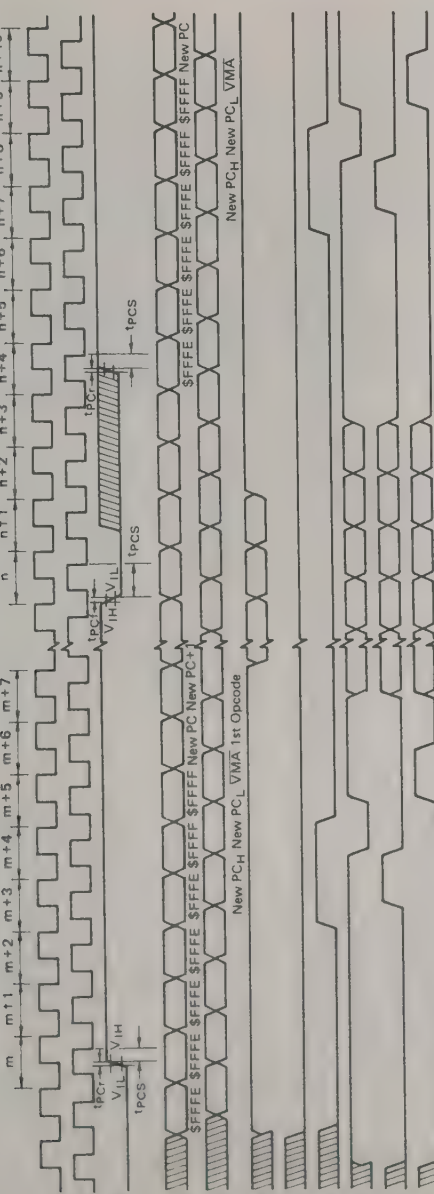
The Bus Status output signal, when decoded with BA, represents the MPU state (valid with leading edge of Q).

MPU State		MPU State Definition
BA	BS	
0	0	Normal (Running)
0	1	Interrupt or RESET Acknowledge
1	0	SYNC Acknowledge
1	1	HALT Acknowledge

Interrupt Acknowledge is indicated during both cycles of a hardware-vector-fetch ($\overline{RES}$, $\overline{NMI}$, $\overline{FIRQ}$, $\overline{IRQ}$, SWI, SWI2, SWI3). This signal, plus decoding of the lower four address lines, can provide the user with an indication of which interrupt level is being serviced and allow vectoring by device. See Table 1.

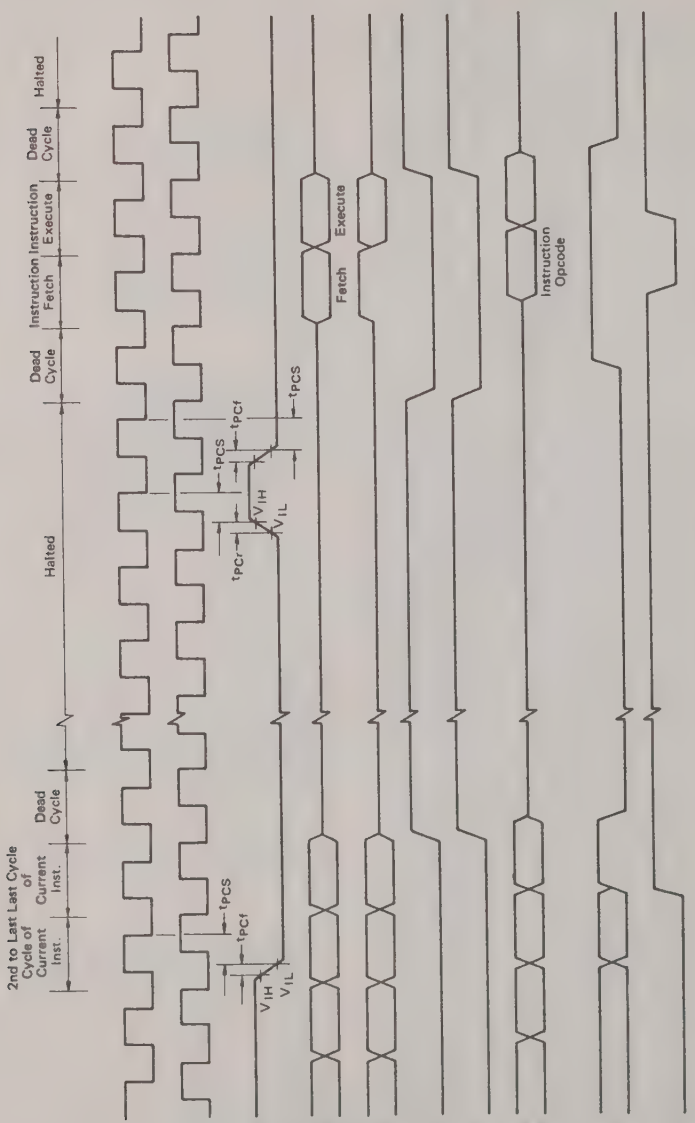
Sync Acknowledge is indicated while the MPU is waiting for external synchronization on an interrupt line.

Halt Acknowledge is indicated when the HD6809E is in a Halt condition.



(NOTE) Waveform measurements for all inputs and outputs are specified at logic "High" = V_{IHmin} and logic "Low" = V_{ILmax} unless otherwise specified.

Figure 7 $\overline{RES}$ Timing



(NOTE) Waveform measurements for all inputs and outputs are specified at logic "High" = V_{IHmin} and logic "Low" = V_{ILmax} unless otherwise specified.

Figure 8 HALT and Single Instruction Execution for System Debug

- **Non Maskable Interrupt ($\overline{NMI}$)***

A negative transition on this input requests that a non-maskable interrupt sequence be generated. A non-maskable interrupt cannot be inhibited by the program, and also has a higher priority than $\overline{FIRQ}$, $\overline{IRQ}$ or software interrupts. During recognition of an $\overline{NMI}$, the entire machine state is saved on the hardware stack. After reset, an $\overline{NMI}$ will not be recognized until the first program load of the Hardware Stack Pointer (S). The pulse width of $\overline{NMI}$ low must be at least one E cycle. If the $\overline{NMI}$ input does not meet the minimum set up with respect to Q, the interrupt will not be recognized until the next cycle. See Figure 9.

- **Fast-Interrupt Request ($\overline{FIRQ}$)***

A "Low" level on this input pin will initiate a fast interrupt sequence, provided its mask bit (F) in the CC is clear. This sequence has priority over the standard Interrupt Request ($\overline{IRQ}$), and is fast in the sense that it stacks only the contents of the condition code register and the program counter. The interrupt service routine should clear the source of the interrupt before doing an RTI. See Figure 10.

- **Interrupt Request ($\overline{IRQ}$)***

A "Low" level input on this pin will initiate an Interrupt Request sequence provided the mask bit (I) in the CC is clear. Since $\overline{IRQ}$ stacks the entire machine state it provides a slower response to interrupts than $\overline{FIRQ}$. $\overline{IRQ}$ also has a lower priority than $\overline{FIRQ}$. Again, the interrupt service routine should clear the source of the interrupt before doing an RTI. See Figure 9.

* $\overline{NMI}$, $\overline{FIRQ}$, and $\overline{IRQ}$ requests are sampled on the falling edge of Q. One cycle is required for synchronization before these interrupts are recognized. The pending interrupt(s) will not be serviced until completion of the current instruction unless a SYNC or CWAI condition is present. If $\overline{IRQ}$ and $\overline{FIRQ}$ do not remain "Low" until completion of the current instruction they may not be recognized. However, $\overline{NMI}$ is latched and need only remain "Low" for one cycle.

- **Clock Inputs E, Q**

E and Q are the clock signals required by the HD6809E. Q must lead E; that is, a transition on Q must be followed by a similar transition on E after a minimum delay. Addresses will be valid from the MPU, t_{AD} after the falling edge of E, and data will be latched from the bus by the falling edge of E. While the Q input is fully TTL compatible, the E input directly drives internal MOS circuitry and, thus, requires levels above normal TTL levels. This approach minimizes clock skew inherent with an internal buffer. Timing and waveforms for E and Q are shown in Figures 1 and 2 while Figure 11 shows a simple clock generator for the HD6809E.

- **BUSY**

Busy will be "High" for the read and modify cycles of a read-modify-write instruction and during the access of the first byte of a double-byte operation (e.g., LDX, STD, ADDD). Busy is also "High" during the first byte of any indirect or other vector fetch (e.g., jump extended, SWI indirect etc.).

In a multi-processor system, busy indicates the need to defer the rearbitration of the next bus cycle to insure the integrity of the above operations. This difference provides the indivisible memory access required for a "test-and-set" primitive, using any one of several read-modify-write instructions.

Busy does not become active during PSH or PUL operations. A typical read-modify-write instruction (ASL) is shown in Figure 12. Timing information is given in Figure 13. Busy is valid t_{CD} after the rising edge of Q.

- **AVMA**

AVMA is the Advanced VMA signal and indicates that the MPU will use the bus in the following bus cycle. The predictive nature of the AVMA signal allows efficient shared-bus multi-processor systems. AVMA is "Low" when the MPU is in either a HALT or SYNC state. AVMA is valid t_{CD} after the rising edge of Q.

- **LIC**

LIC (Last Instruction Cycle) is "High" during the last cycle of every instruction, and its transition from "High" to "Low" will indicate that the first byte of an opcode will be latched at the end of the present bus cycle. LIC will be "High" when the MPU is Halted at the end of an instruction, (i.e., not in CWAI or RESET) in SYNC state or while stacking during interrupts. LIC is valid t_{CD} after the rising edge of Q.

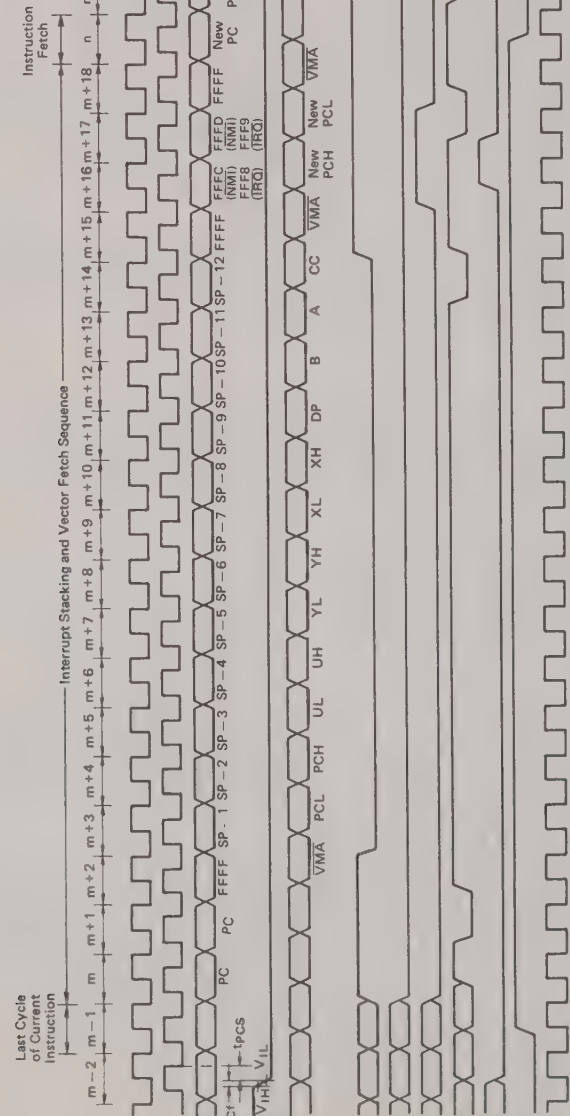
- **TSC**

TSC (Three-State Control) will cause MOS address, data, and R/$\overline{W}$ buffers to assume a high-impedance state. The control signals (BA, BS, BUSY, AVMA and LIC) will not go to the high-impedance state. TSC is intended to allow a single bus to be shared with other bus masters (processors or DMA controllers).

While E is "Low", TSC controls the address buffers and R/$\overline{W}$ directly. The data bus buffers during a write operation are in a high-impedance state until Q rises at which time, if TSC is true, they will remain in a high-impedance state. If TSC is held beyond the rising edge of E, then it will be internally latched, keeping the bus drivers in a high-impedance state for the remainder of the bus cycle. See Figure 14.

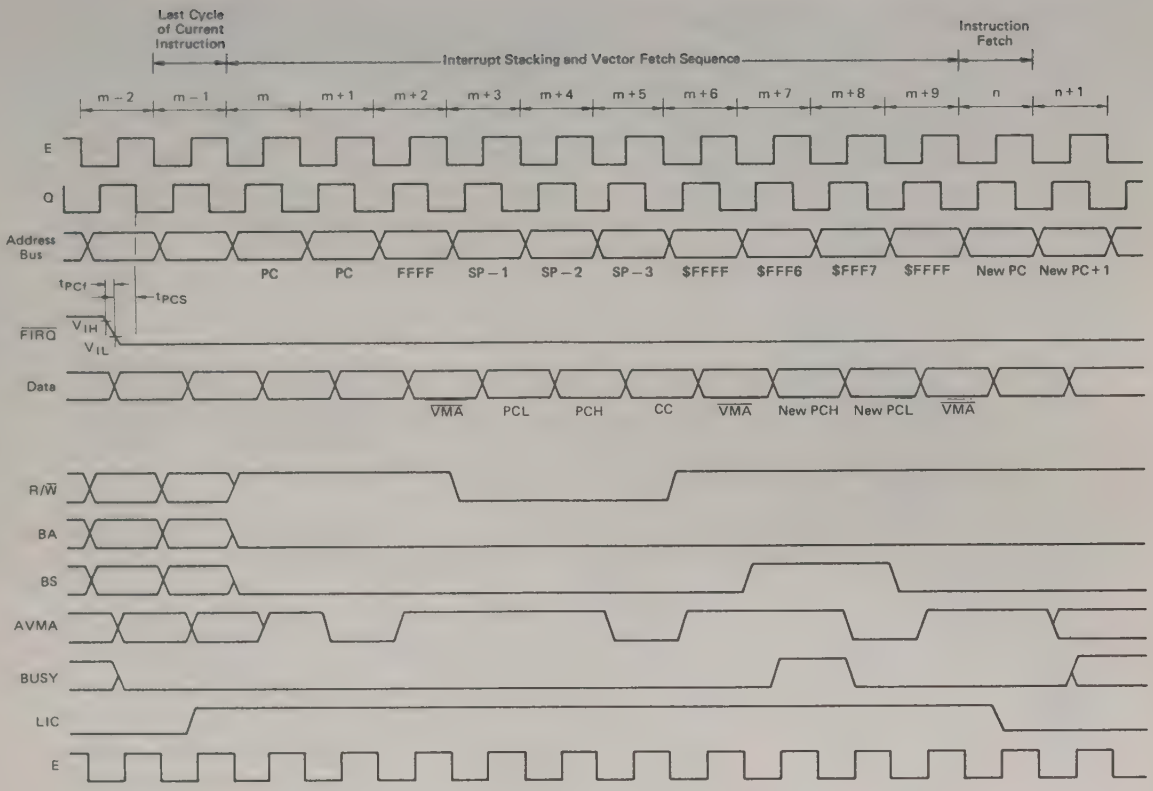
- **MPU Operation**

During normal operation, the MPU fetches an instruction from memory and then executes the requested function. This sequence begins after $\overline{RES}$ and is repeated indefinitely unless altered by a special instruction or hardware occurrence. Software instructions that alter normal MPU operation are: SWI, SWI2, SWI3, CWAI, RTI and SYNC. An interrupt or $\overline{HALT}$ input can also alter the normal execution of instructions. Figure 15 illustrates the flow chart for the HD6809E.


Last Cycle of Current Instruction
Instruction Fetch
Interrupt Stacking and Vector Fetch Sequence
m−2 m−1 m m+1 m+2 m+3 m+4 m+5 m+6 m+7 m+8 m+9 m+10 m+11 m+12 m+13 m+14 m+15 m+16 m+17 m+18 n n+1
E
Q
Address Bus
PC PC FFFF SP−1 SP−2 SP−3 SP−4 SP−5 SP−6 SP−7 SP−8 SP−9 SP−10 SP−11 SP−12 FFFF FFFC (NMI) FFF8 (IRQ) FFFD (NMI) FFF9 (IRQ) FFFF New PC New PC+1
tPCf
tPCS
IRQ or NMI
VIH
VIL
Data
VMA PCL PCH UL UH YL YH XL XH DP B A CC VMA New PCH New PCL VMA
R/W
BA
BS
AVMA
BUSY
LIC
E

(NOTE) Waveform measurements for all inputs and outputs are specified at logic "High" = V_{IHmin} and logic "Low" = V_{ILmax} unless otherwise specified.
E clock shown for reference only.

Figure 9 $\overline{IRQ}$ and $\overline{NMI}$ Interrupt Timing



(NOTE) Waveform measurements for all inputs and outputs are specified at logic "High" = V_{IHmin} and logic "Low" = V_{ILmax} unless otherwise specified.
E clock shown for reference only.

Figure 10 $\overline{\text{FIRQ}}$ Interrupt Timing

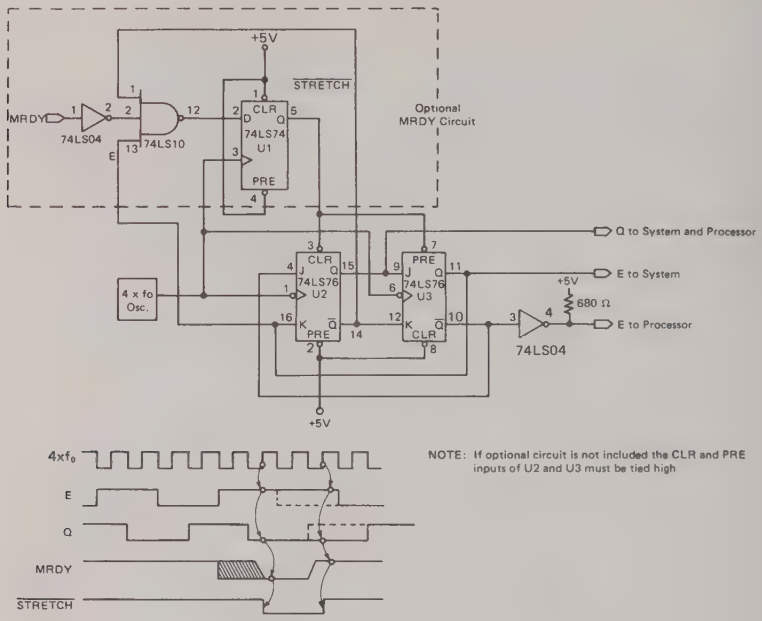


Figure 11 HD6809E Clock Generator

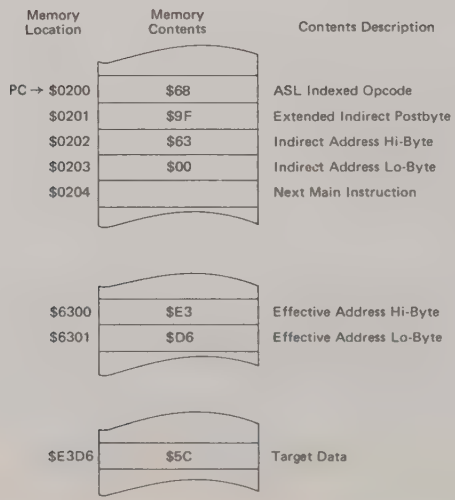
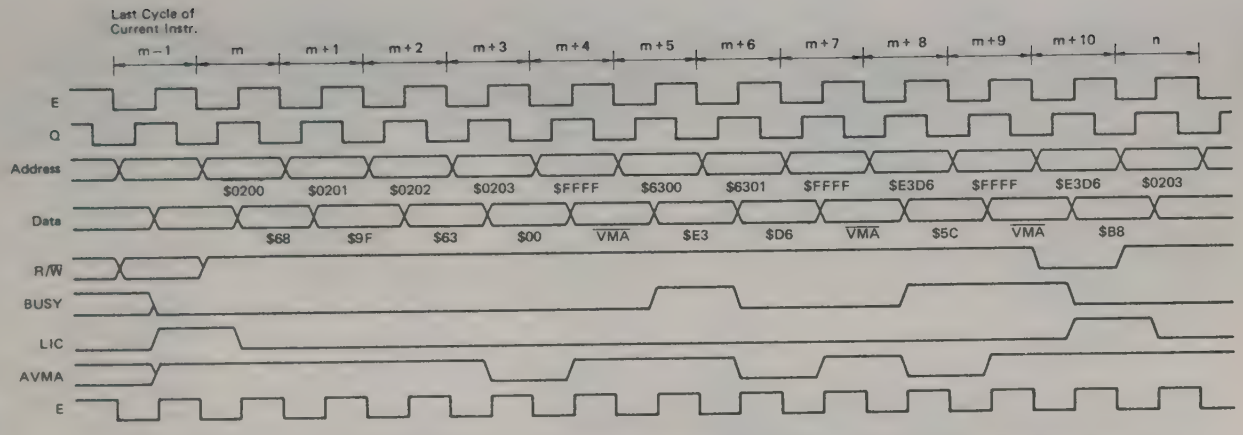
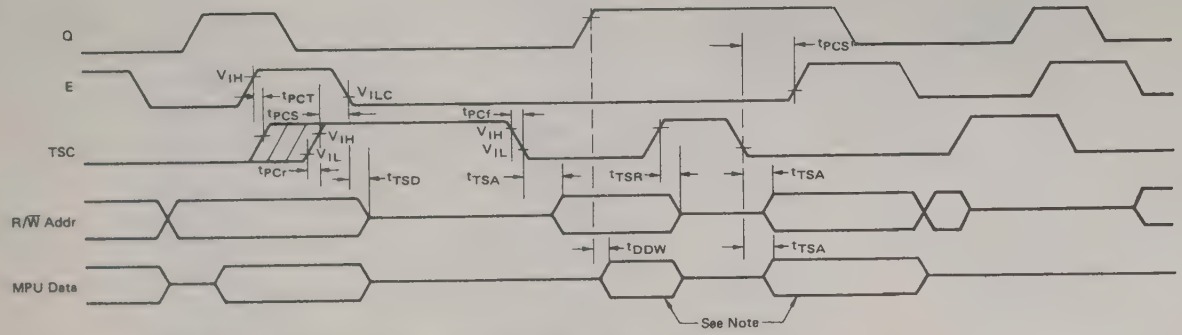


Figure 12 Read Modify Write Instruction Example (ASL Extended Indirect)



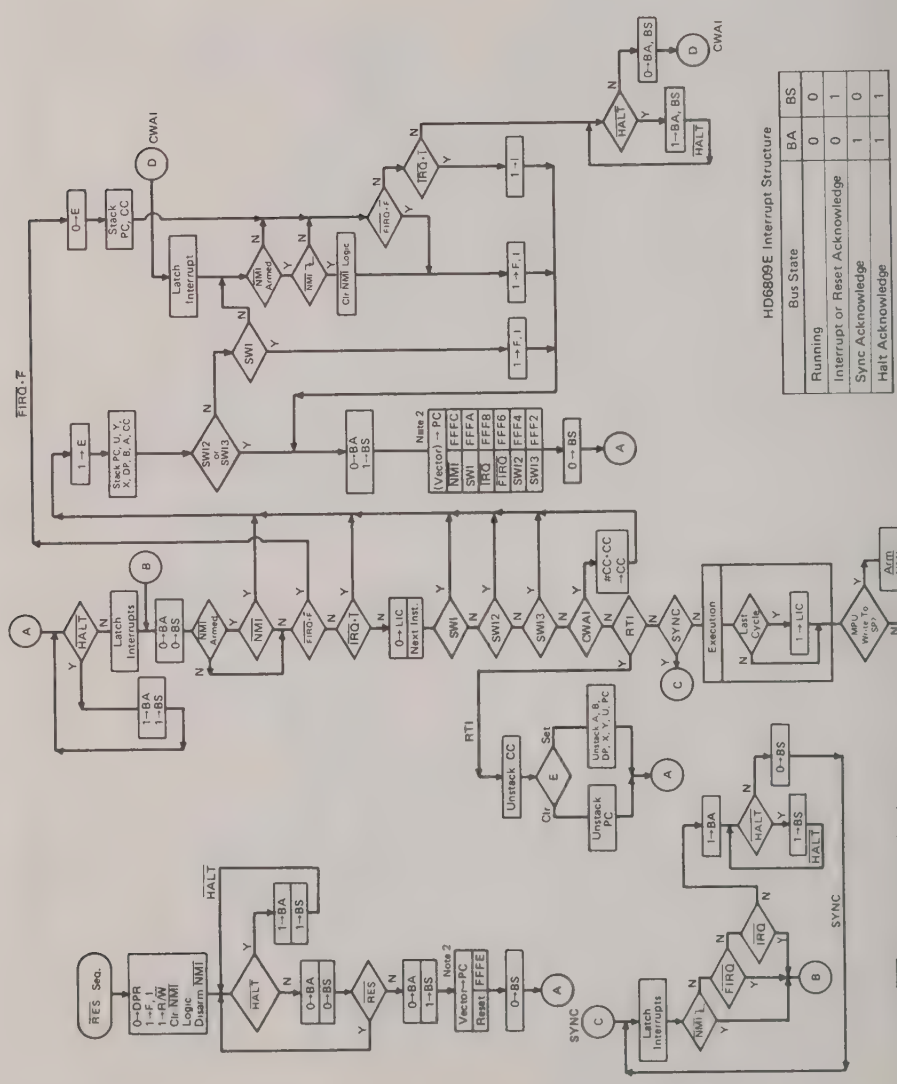
(NOTE) Waveform measurements for all inputs and outputs are specified at logic "High" = V_{IHmin} and logic "Low" = V_{ILmax} unless otherwise specified.

Figure 13 BUSY Timing (ASL Extended Indirect Instruction)



(NOTES) Data will be asserted by the MPU only during the interval while R/W is "Low" and E or Q is "High".
Waveform measurements for all inputs and outputs are specified at logic "High" = V_{IHmin} and logic "Low" = V_{ILmax} unless otherwise specified.

Figure 14 TSC Timing



HD6809E Interrupt Structure

Bus State	BA	BS
Running	0	0
Interrupt or Reset Acknowledge	0	1
Sync Acknowledge	1	0
Halt Acknowledge	1	1

Figure 15 Flowchart for HD6809E Instruction

(NOTES) 1. Asserting $\overline{RES}$ will result in entering the reset sequence from any point in the flow chart.
2. BUSY is "High" during first vector fetch cycle.

■ ADDRESSING MODES

The basic instructions of any computer are greatly enhanced by the presence of powerful addressing modes. The HD6809E has the most complete set of addressing modes available on any microcomputer today. For example, the HD6809E has 59 basic instructions; however, it recognizes 1464 different variations of instructions and addressing modes. The addressing modes support modern programming techniques. The following addressing modes are available on the HD6809E:

(1) Implied (Includes Accumulator)
(2) Immediate
(3) Extended
(4) Extended Indirect
(5) Direct
(6) Register
(7) Indexed
    - Zero-Offset
    - Constant Offset
    - Accumulator Offset
    - Auto Increment/Decrement
(8) Indexed Indirect
(9) Relative
(10) Program Counter Relative

• Implied (Includes Accumulator)

In this addressing mode, the opcode of the instruction contains all the address information necessary. Examples of Implied Addressing are: ABX, DAA, SWI, ASRA, and CLRB.

• Immediate Addressing

In Immediate Addressing, the effective address of the data is the location immediately following the opcode (i.e., the data to be used in the instruction immediately follows the opcode of the instruction). The HD6809E uses both 8 and 16-bit immediate values depending on the size of argument specified by the opcode. Examples of instructions with immediate Addressing are:

```
LDA  #$20
LDX  #$F000
LDY  #CAT
```

(NOTE) # signifies immediate addressing, $ signifies hexadecimal value.

• Extended Addressing

In Extended Addressing, the contents of the two bytes immediately following the opcode fully specify the 16-bit effective address used by the instruction. Note that the address generated by an extended instruction defines an absolute address and is not position independent. Examples of Extended Addressing include:

```
LDA  CAT
STX  MOUSE
LDD  $2000
```

• Extended Indirect

As a special case of indexed addressing (discussed below), one level of indirection may be added to Extended Addressing. In Extended Indirect, the two bytes following the postbyte of an Indexed instruction contain the address of the data.

```
LDA  [CAT]
LDX  [$FFFE]
STU  [DOG]
```

• Direct Addressing

Direct addressing is similar to extended addressing except that only one byte of address follows the opcode. This byte specifies the lower 8 bits of the address to be used. The upper 8 bits of the address are supplied by the direct page register. Since only one byte of address is required in direct addressing, this mode requires less memory and executes faster than extended addressing. Of course, only 256 locations (one page) can be accessed without redefining the contents of the DP register. Since the DP register is set to $00 on Reset, direct addressing on the HD6809E is compatible with direct addressing on the HD6800. Indirection is not allowed in direct addressing. Some examples of direct addressing are:

```
LDA   $30
SETDP $10     (Assembler directive)
LDB   $1030
LDD   <CAT
```

(NOTE) < is an assembler directive which forces direct addressing.

• Register Addressing

Some opcodes are followed by a byte that defines a register or set of registers to be used by the instruction. This is called a postbyte. Some examples of register addressing are:

```
TFR   X, Y        Transfer X into Y
EXG   A, B        Exchanges A with B
PSHS  A, B, X, Y  Push Y, X, B and A onto S
PULU  X, Y, D     Pull D, X, and Y from U
```

• Indexed Addressing

In all indexed addressing, one of the pointer registers (X, Y, U, S, and sometimes PC) is used in a calculation of the effective address of the operand to be used by the instruction. Five basic types of indexing are available and are discussed below. The postbyte of an indexed instruction specifies the basic type and variation of the addressing mode as well as the pointer register to be used. Figure 16 lists the legal formats for the postbyte. Table 2 gives the assembler form and the number of cycles and bytes added to the basic values for indexed addressing for each variation.

Post-Byte Register Bit								Indexed Addressing Mode
7	6	5	4	3	2	1	0	
0	R	R	d	d	d	d	d	EA = ,R + 5 Bit Offset
1	R	R	0	0	0	0	0	,R +
1	R	R	1	0	0	0	1	,R + +
1	R	R	0	0	0	1	0	, -R
1	R	R	1	0	0	1	1	, - - R
1	R	R	1	0	1	0	0	EA = ,R + 0 Offset
1	R	R	1	0	1	0	1	EA = ,R + ACCB Offset
1	R	R	1	0	1	1	0	EA = ,R + ACCA Offset
1	R	R	1	1	0	0	0	EA = , R + 8 Bit Offset
1	R	R	1	1	0	0	1	EA = ,R + 16 Bit Offset
1	R	R	1	1	0	1	1	EA = ,R + D Offset
1	x	x	1	1	1	0	0	EA = ,PC + 8 Bit Offset
1	x	x	1	1	1	0	1	EA = ,PC + 16 Bit Offset
1	R	R	1	1	1	1	1	EA = [,Address]

Addressing Mode Field
Indirect Field (Sign bit when b7 = 0)
Register Field RR
00 = X
01 = Y
10 = U
11 = S

x = Don't Care
d = Offset Bit
i = 0 = Non Indirect, 1 = Indirect

Figure 16 Index Addressing Postbyte Register Bit Assignments

Table 2 Indexed Addressing Mode

Type	Forms	Non Indirect				Indirect			
		Assembler Form	Postbyte OP Code	+ ~	+ #	Assembler Form	Postbyte OP Code	+ ~	+ #
Constant Offset From R (2's Complement Offsets)	No Offset	,R	1RR00100	0	0	[,R]	1RR10100	3	0
	5 Bit Offset	n, R	0RRnnnnn	1	0	defaults to 8-bit			
	8 Bit Offset	n, R	1RR01000	1	1	[n, R]	1RR11000	4	1
	16 Bit Offset	n, R	1RR01001	4	2	[n, R]	1RR11001	7	2
Accumulator Offset From R (2's Complement Offsets)	A Register Offset	A, R	1RR00110	1	0	[A, R]	1RR10110	4	0
	B Register Offset	B, R	1RR00101	1	0	[B, R]	1RR10101	4	0
	D Register Offset	D, R	1RR01011	4	0	[D, R]	1RR11011	7	0
Auto Increment/Decrement R	Increment By 1	,R +	1RR00000	2	0	not allowed			
	Increment By 2	,R + +	1RR00001	3	0	[,R + +]	1RR10001	6	0
	Decrement By 1	, - R	1RR00010	2	0	not allowed			
	Decrement By 2	, - - R	1RR00011	3	0	[, - - R]	1RR10011	6	0
Constant Offset From PC (2's Complement Offsets)	8 Bit Offset	n, PCR	1xx01100	1	1	[n, PCR]	1xx11100	4	1
	16 Bit Offset	n, PCR	1xx01101	5	2	[n, PCR]	1xx11101	8	2
Extended Indirect	16 Bit Address	–	–	–	–	[n]	10011111	5	2

R = X, Y, U or S
x = Don't Care

RR:
00 = X
01 = Y
10 = U
11 = S

$\overset{+}{\sim}$ and $\overset{+}{\#}$ indicate the number of additional cycles and bytes for the particular variation.

Zero-Offset Indexed

In this mode, the selected pointer register contains the effective address of the data to be used by the instruction. This is the fastest indexing mode.

Examples are:

```
LDD   0, X
LDA   S
```

Constant Offset Indexed

In this mode, a two's-complement offset and the contents of one of the pointer registers are added to form the effective address of the operand. The pointer register's initial content is unchanged by the addition.

Three sizes of offsets are available:

5-bit (−16 to +15)
8-bit (−128 to +127)
16-bit (−32768 to +32767)

The two's complement 5-bit offset is included in the postbyte and, therefore, is most efficient in use of bytes and cycles. The two's complement 8-bit offset is contained in a single byte following the postbyte. The two's complement 16-bit offset is in the two bytes following the postbyte. In most cases the programmer need not be concerned with the size of this offset since the assembler will select the optimal size automatically.

Examples of constant-offset indexing are:

```
LDA   23, X
LDX   −2, S
LDY   300, X
LDU   CAT, Y
```

Accumulator-Offset Indexed

This mode is similar to constant offset indexed except that the two's-complement value in one of the accumulators (A, B or D) and the contents of one of the pointer registers are added to form the effective address of the operand. The contents of both the accumulator and the pointer register are unchanged by the addition. The postbyte specifies which accumulator to use as an offset and no additional bytes are required. The advantage of an accumulator offset is that the value of the offset can be calculated by a program at run-time.

Some examples are:

```
LDA   B, Y
LDX   D, Y
LEAX  B, X
```

Auto Increment/Decrement Indexed

In the auto increment addressing mode, the pointer register contains the address of the operand. Then, after the pointer register is used it is incremented by one or two. This addressing mode is useful in stepping through tables, moving data, or for the creation of software stacks. In auto decrement, the pointer register is decremented prior to use as the address of the data. The use of auto decrement is similar to that of auto increment; but the tables, etc., are scanned from the high to low addresses. The size of the increment/decrement can be either one or two to allow for tables of either 8- or 16-bit data to be accessed and is selectable by the programmer. The pre-

decrement, post-increment nature of these modes allow them to be used to create additional software stacks that behave identically to the U and S stacks.

Some examples of the auto increment/decrement addressing modes are:

```
LDA   ,X+
STD   ,Y++
LDB   ,-Y
LDX   ,--S
```

Care should be taken in performing operations on 16-bit pointer registers (X, Y, U, S) where the same register is used to calculate the effective address.

Consider the following instruction:

STX 0, X++ (X initialized to 0)

The desired result is to store a 0 in locations $0000 and $0001 then increment X to point to $0002. In reality, the following occurs:

0 → temp	calculate the EA; temp is a holding register
X + 2 → X	perform autoincrement
X → (temp)	do store operation

- **Indexed Indirect**

All of the indexing modes with the exception of auto increment/decrement by one, or a ±4-bit offset may have an additional level of indirection specified. In indirect addressing, the effective address is contained at the location specified by the contents of the Index Register plus any offset. In the example below, the A accumulator is loaded indirectly using an effective address calculated from the Index Register and an offset.

```
          Before Execution
          A = XX (don't care)
          X = $F000
$0100     LDA [$10, X]      EA is now $F010
$F010     $F1               $F150 is now the
$F011     $50               new EA
$F150     $AA
          After Execution
          A = $AA (Actual Data Loaded)
          X = $F000
```

All modes of indexed indirect are included except those which are meaningless (e.g., auto increment/decrement by 1 indirect). Some examples of indexed indirect are:

```
LDA   [,X]
LDD   [10,S]
LDA   [B,Y]
LDD   [,X++]
```

- **Relative Addressing**

The byte(s) following the branch opcode is (are) treated as a signed offset which may be added to the program counter. If the branch condition is true then the calculated address (PC + signed offset) is loaded into the program counter. Program execution continues at the new location as indicated by the PC; short (1 byte offset) and long (2 bytes offset) relative addressing modes are available. All of memory can be reached in long relative addressing as an effective address is interpreted modulo 2^{16}. Some examples of relative addressing are:

```
BEQ     CAT     (short)
BGT     DOG     (short)

CAT     LBEQ    RAT     (long)
DOG     LBGT    RABBIT  (long)
        .
        .
        .
RAT     NOP
RABBIT  NOP
```

- **Program Counter Relative**

The PC can be used as the pointer register with 8 or 16-bit signed offsets. As in relative addressing, the offset is added to the current PC to create the effective address. The effective address is then used as the address of the operand or data. Program Counter Relative Addressing is used for writing position independent programs. Tables related to a particular routine will maintain the same relationship after the routine is moved, if referenced relative to the Program Counter. Examples are:

```
LDA   CAT,PCR
LEAX  TABLE,PCR
```

Since program counter relative is a type of indexing, an additional level of indirection is available.

```
LDA   [CAT,PCR]
LDU   [DOG,PCR]
```

■ HD6809E INSTRUCTION SET

The instruction set of the HD6809E is similar to that of the HD6800 and is upward compatible at the source code level. The number of opcodes has been reduced from 72 to 59, but because of the expanded architecture and additional addressing modes, the number of available opcodes (with different addressing modes) has risen from 197 to 1464.

Some of the new instructions are described in detail below:

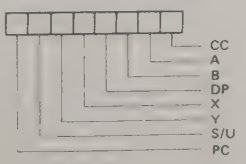
- **PSHU/PSHS**

The push instructions have the capability of pushing onto either the hardware stack (S) or user stack (U) any single register, or set of registers with a single instruction.

- **PULU/PULS**

The pull instructions have the same capability of the push instruction, in reverse order. The byte immediately following the push or pull opcode determines which register or registers are to be pushed or pulled. The actual PUSH/PULL sequence is fixed; each bit defines a unique register to push or pull, as shown in below.

PUSH/PULL POST BYTE



	← Pull Order					Push Order →	
PC	U	Y	X	DP	B	A	CC
FFFF	← increasing memory address						0000
PC	S	Y	X	DP	B	A	CC

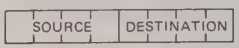
- **TFR/EXG**

Within the HD6809E, any register may be transferred to or exchanged with another of like-size; i.e., 8-bit to 8-bit or 16-bit to 16-bit. Bits 4~7 of postbyte define the source register, while bits 0~3 represent the destination register. These are denoted as follows:

0000 – D	0101 – PC
0001 – X	1000 – A
0010 – Y	1001 – B
0011 – U	1010 – CC
0100 – S	1011 – DP

(NOTE) All other combinations are undefined and INVALID.

TRANSFER/EXCHANGE POST BYTE



- **LEAX/LEAY/LEAU/LEAS**

The LEA (Load Effective Address) works by calculating the effective address used in an indexed instruction and stores that address value, rather than the data at that address, in a pointer register. This makes all the features of the internal addressing hardware available to the programmer. Some of the implications of this instruction are illustrated in Table 3.

The LEA instruction also allows the user to access data in a position independent manner. For example:

```
        LEAX    MSG1, PCR
        LBSR    PDATA (Print message routine)
        .
        .
MSG1    FCC     'MESSAGE'
```

This sample program prints: 'MESSAGE'. By writing MSG1, PCR, the assembler computes the distance between the present address and MSG1. This result is placed as a constant into the LEAX instruction which will be indexed from the PC value at the time of execution. No matter where the code is located, when it is executed, the computed offset from the PC will put the absolute address of MSG1 into the X pointer register. This code is totally position independent.

The LEA instructions are very powerful and use an internal holding register (temp). Care must be exercised when using the LEA instructions with the autoincrement and autodecrement addressing modes due to the sequence of internal operations. The LEA internal sequence is outlined as follows:

```
LEAa, b+        (any of the 16-bit pointer registers X, Y, U
                 or S may be substituted for a and b.)
1. b → temp     (calculate the EA)
2. b + 1 → b    (modify b, postincrement)
3. temp → a     (load a)

LEAa, – b
1. b – 1 → temp (calculate EA with predecrement)
2. b – 1 → b    (modify b, predecrement)
3. temp → a     (load a)
```

Autoincrement-by-two and autodecrement-by-two instructions work similarly. Note that LEAX, X+ does not change X, however LEAX, –X does decrement X. LEAX 1, X should be used to increment X by one.

Table 3 LEA Examples

Instruction	Operation	Comment
LEAX 10, X	X + 10 → X	Adds 5-bit constant 10 to X
LEAX 500, X	X + 500 → X	Adds 16-bit constant 500 to X
LEAY A, Y	Y + A → Y	Adds 8-bit A accumulator to Y
LEAY D, Y	Y + D → Y	Adds 16-bit D accumulator to Y
LEAU –10, U	U – 10 → U	Subtracts 10 from U
LEAS –10, S	S – 10 → S	Used to reserve area on stack
LEAS 10, S	S + 10 → S	Used to 'clean up' stack
LEAX 5, S	S + 5 → X	Transfers as well as adds

- **MUL**

Multiplies the unsigned binary numbers in the A and B accumulator and places the unsigned result into the 16-bit D accumulator. This unsigned multiply also allows multiple-precision multiplications.

Long and Short Relative Branches

The HD6809E has the capability of program counter relative branching throughout the entire memory map. In this mode, if the branch is to be taken, the 8 or 16-bit signed offset is added to the value of the program counter to be used as the effective address. This allows the program to branch anywhere in the 64k memory map. Position independent code can be easily generated through the use of relative branching. Both short (8-bit) and long (16-bit) branches are available.

- **SYNC**

After encountering a Sync instruction, the MPU enters a Sync state, stops processing instructions and waits for an interrupt. If the pending interrupt is non-maskable ($\overline{NMI}$) or maskable ($\overline{FIRQ}$, $\overline{IRQ}$) with its mask bit (F or I) clear, the processor will clear the Sync state and perform the normal interrupt stacking and service routine. Since $\overline{FIRQ}$ and $\overline{IRQ}$ are not edge-triggered, a low level with a minimum duration of three bus cycles is required to assure that the interrupt will be taken. If the pending interrupt is maskable ($\overline{FIRQ}$, $\overline{IRQ}$) with its mask bit (F or I) set, the processor will clear the Sync state and continue processing by executing the next inline instruction. Figure 17 depicts Sync timing.

Software Interrupts

A Software Interrupt is an instruction which will cause an interrupt, and its associated vector fetch. These Software Interrupts are useful in operating system calls, software debugging, trace operations, memory mapping, and software development systems. Three levels of SWI are available on this HD6809E, and are prioritized in the following order: SWI, SWI2, SWI3.

16-Bit Operation

The HD6809E has the capability of processing 16-bit data. These instructions include loads, stores, compares, adds, subtracts, transfers, exchanges, pushes and pulls.

- **CYCLE-BY-CYCLE OPERATION**

The address bus cycle-by-cycle performance chart illustrates the memory-access sequence corresponding to each possible instruction and addressing mode in the HD6809E. Each instruction begins with an opcode fetch. While that opcode is being internally decoded, the next program byte is always fetched. (Most instructions will use the next byte, so this

technique considerably speeds throughput.) Next, the operation of each opcode will follow the flow chart. $\overline{VMA}$ is an indication of $FFFF_{16}$ on the address bus, R/$\overline{W}$ = "High" and BS = "Low". The following examples illustrate the use of the chart; see Figure 18.

Example 1: LBSR (Branch Taken)
Before Execution SP = F000

```
                .
                .
                .
$8000           LBSR    CAT
                .
                .
                .
$A000   CAT     .
```

CYCLE-BY-CYCLE FLOW

Cycle #	Address	Data	R/$\overline{W}$	Description
1	8000	17	1	Opcode Fetch
2	8001	1F	1	Offset High Byte
3	8002	FD	1	Offset Low Byte
4	FFFF	*	1	$\overline{VMA}$ Cycle
5	FFFF	*	1	$\overline{VMA}$ Cycle
6	FFFF	*	1	$\overline{VMA}$ Cycle
7	FFFF	*	1	$\overline{VMA}$ Cycle
8	EFFF	03	0	Stack Low Order Byte of Return Address
9	EFFE	80	0	Stack High Order Byte of Return Address

Example 2: DEC (Extended)

```
$8000   DEC     $A000
$A000   FCB     $80
```

CYCLE-BY-CYCLE FLOW

Cycle #	Address	Data	R/$\overline{W}$	Description
1	8000	7A	1	Opcode Fetch
2	8001	A0	1	Operand Address, High Byte
3	8002	00	1	Operand Address, Low Byte
4	FFFF	*	1	$\overline{VMA}$ Cycle
5	A000	80	1	Read the Data
6	FFFF	*	1	$\overline{VMA}$ Cycle
7	A000	7F	0	Store the Decremented Data

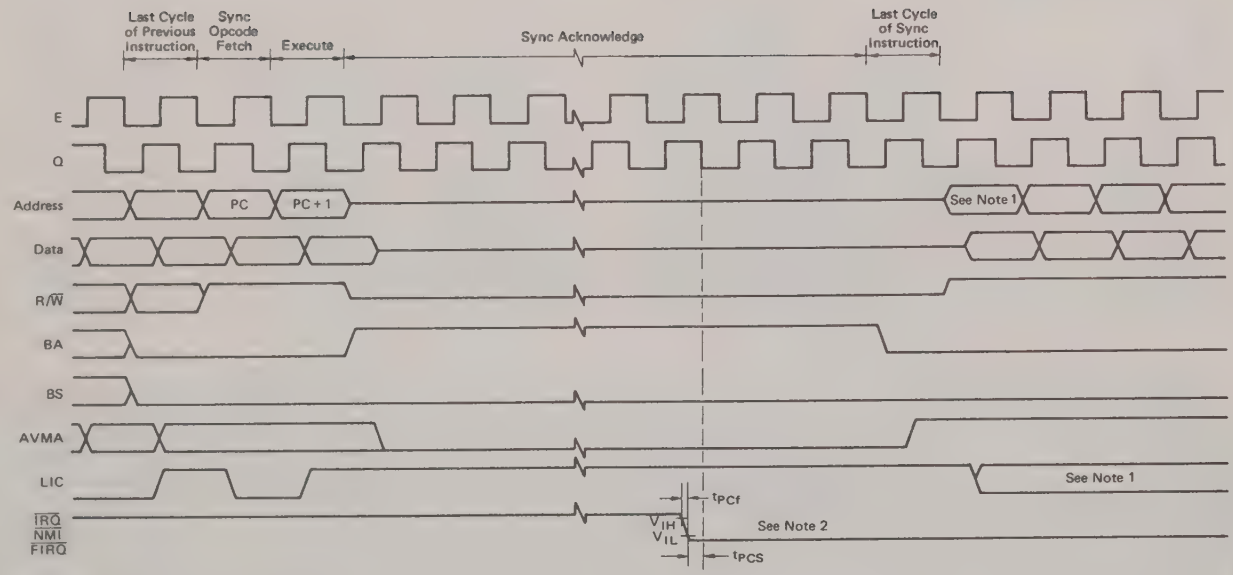
* The data bus has the data at that particular address.

■ HD6809E INSTRUCTION SET TABLES

The instructions of the HD6809E have been broken down into five different categories. They are as follows:

8-Bit operation (Table 4)
16-Bit operation (Table 5)
Index register/stack pointer instructions (Table 6)
Relative branches (long or short) (Table 7)
Miscellaneous instructions (Table 8)

HD6809E instruction set tables and Hexadecimal Values of instructions are shown in Table 9 and Table 10.

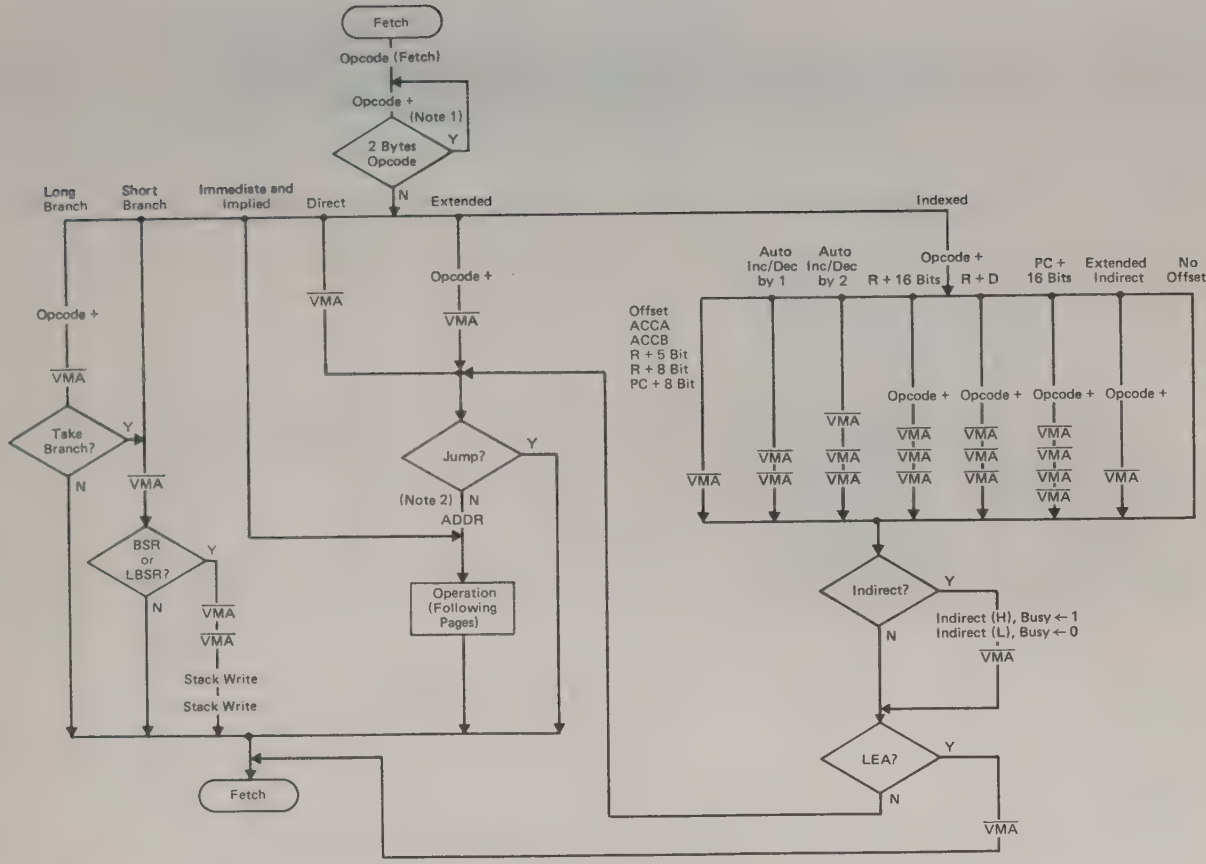


(NOTES) 1. If the associated mask bit is set when the interrupt is requested, LIC will go "Low" and this cycle will be an instruction fetch from address location PC + 1. However, if the interrupt is accepted ($\overline{NMI}$ or an unmasked $\overline{FIRQ}$ or $\overline{IRQ}$) LIC will remain "High" and interrupt processing will start with this cycle as (m) on Figure 9 and 10 (Interrupt Timing).

2. If mask bits are clear, $\overline{IRQ}$ and $\overline{FIRQ}$ must be held "Low" for three cycles to guarantee that interrupt will be taken, although only one cycle is necessary to bring the processor out of SYNC.

3. Waveform measurements for all inputs and outputs are specified at logic "High" = V_{IHmin} and logic "Low" = V_{ILmax} unless otherwise specified.

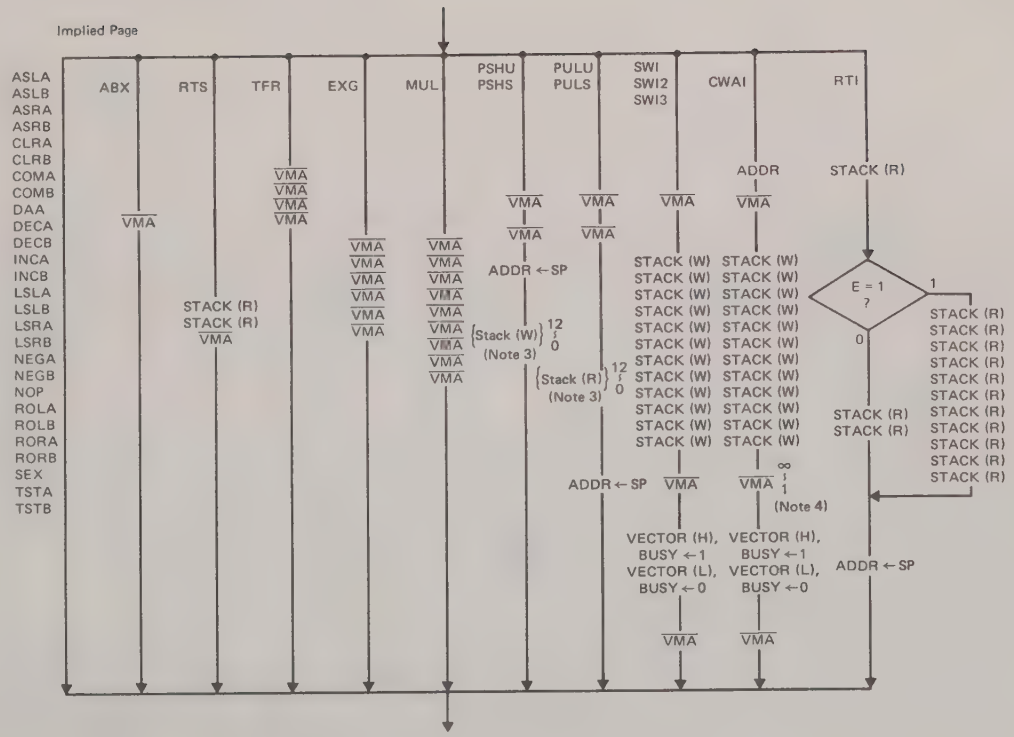
Figure 17 SYNC Timing



(NOTE)

1. Busy = "High" during access of first byte of double byte immediate load.
2. Write operation during store instruction. Busy = "High" during first two cycles of a double-byte access and the first cycle of read-modify-write access.
3. AVMA is asserted on the cycle before a $\overline{\text{VMA}}$ cycle.

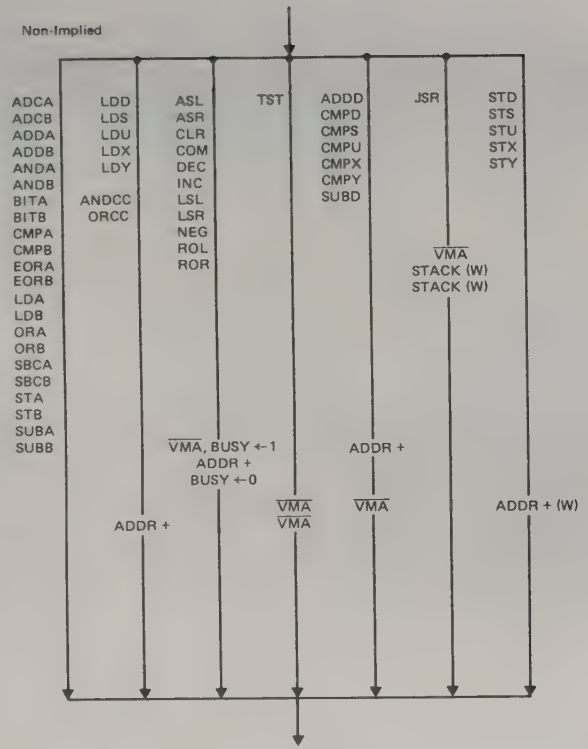
Figure 18 Address Bus Cycle-by-Cycle Performance



(NOTES)

1. Stack (W) refers to the following sequence: SP ← SP − 1, then ADDR ← SP with R/$\overline{W}$ = "Low"
   Stack (R) refers to the following sequence: ADDR ← SP with R/$\overline{W}$ = "High", then SP ← SP + 1.
   PSHU, PULU instructions use the user stack pointer (i.e., SP = U) and PSHS, PULS use the hardware stack pointer (i.e., SP = S).
2. Vector refers to the address of an interrupt or reset vector (see Table 1).
3. The number of stack accesses will vary according to the number of bytes saved.
4. $\overline{VMA}$ cycles will occur until an interrupt occurs.

Figure 18 Address Bus Cycle-by-Cycle Performance (Continued)



(NOTES)

1. Stack (W) refers to the following sequence: SP ← SP − 1, then ADDR ← SP with R/$\overline{W}$ = "Low"
   Stack (R) refers to the following sequence: ADDR ← SP with R/$\overline{W}$ = "High", then SP ← SP + 1.
   PSHU, PULU instructions use the user stack pointer (i.e., SP = U) and PSHS, PULS use the hardware stack pointer (i.e., SP = S).
2. Vector refers to the address of an interrupt or reset vector (see Table 1).
3. The number of stack accesses will vary according to the number of bytes saved.
4. $\overline{VMA}$ cycles will occur until an interrupt occurs.

Figure 18 Address Bus Cycle-by-Cycle Performance (Continued)

Table 4 8-Bit Accumulator and Memory Instructions

Mnemonic(s)	Operation
ADCA, ADCB	Add memory to accumulator with carry
ADDA, ADDB	Add memory to accumulator
ANDA, ANDB	And memory with accumulator
ASL, ASLA, ASLB	Arithmetic shift of accumulator or memory left
ASR, ASRA, ASRB	Arithmetic shift of accumulator or memory right
BITA, BITB	Bit test memory with accumulator
CLR, CLRA, CLRB	Clear accumulator or memory location
CMPA, CMPB	Compare memory from accumulator
COM, COMA, COMB	Complement accumultor or memory location
DAA	Decimal adjust A accumulator
DEC, DECA, DECB	Decrement accumulator or memory location
EORA, EORB	Exclusive or memory with accumulator
EXG R1, R2	Exchange R1 with R2 (R1, R2 = A, B, CC, DP)
INC, INCA, INCB	Increment accumulator or memory location
LDA, LDB	Load accumulator from memory
LSL, LSLA, LSLB	Logical shift left accumulator or memory location
LSR, LSRA, LSRB	Logical shift right accumulator or memory location
MUL	Unsigned multiply (A × B → D)
NEG, NEGA, NEGB	Negate accumulator or memory
ORA, ORB	Or memory with accumulator
ROL, ROLA, ROLB	Rotate accumulator or memory left
ROR, RORA, RORB	Rotate accumulator or memory right
SBCA, SBCB	Subtract memory from accumulator with borrow
STA, STB	Store accumulator to memory
SUBA, SUBB	Subtract memory from accumulator
TST, TSTA, TSTB	Test accumulator or memory location
TFR R1, R2	Transfer R1 to R2 (R1, R2 = A, B, CC, DP)

(NOTE) A, B, CC or DP may be pushed to (pulled from) either stack with PSHS, PSHU (PULS, PULU) instructions.

Table 5 16-Bit Accumulator and Memory Instructions

Mnemonic(s)	Operation
ADDD	Add memory to D accumulator
CMPD	Compare memory from D accumulator
EXG D, R	Exchange D with X, Y, S, U or PC
LDD	Load D accumulator from memory
SEX	Sign Extend B accumulator into A accumulator
STD	Store D accumulator to memory
SUBD	Subtract memory from D accumulator
TFR D, R	Transfer D to X, Y, S, U or PC
TFR R, D	Transfer X, Y, S, U or PC to D

(NOTE) D may be pushed (pulled) to either stack with PSHS, PSHU (PULS, PULU) instructions.

Table 6 Index Register Stack Pointer Instructions

Mnemonic(s)	Operation
CMPS, CMPU	Compare memory from stack pointer
CMPX, CMPY	Compare memory from index register
EXG R1, R2	Exchange D, X, Y, S, U or PC with D, X, Y, S, U or PC
LEAS, LEAU	Load effective address into stack pointer
LEAX, LEAY	Load effective address into index register
LDS, LDU	Load stack pointer from memory
LDX, LDY	Load index register from memory
PSHS	Push A, B, CC, DP, D, X, Y, U, or PC onto hardware stack
PSHU	Push A, B, CC, DP, D, X, Y, S, or PC onto user stack
PULS	Pull A, B, CC, DP, D, X, Y, U or PC from hardware stack
PULU	Pull A, B, CC, DP, D, X, Y, S or PC from user stack
STS, STU	Store stack pointer to memory
STX, STY	Store index register to memory
TFR R1, R2	Transfer D, X, Y, S, U or PC to D, X, Y, S, U or PC
ABX	Add B accumulator to X (unsigned)

Table 7 Branch Instructions

Mnemonic(s)	Operation
	SIMPLE BRANCHES
BEQ, LBEQ	Branch if equal
BNE, LBNE	Branch if not equal
BMI, LBMI	Branch if minus
BPL, LBPL	Branch if plus
BCS, LBCS	Branch if carry set
BCC, LBCC	Branch if carry clear
BVS, LBVS	Branch if overflow set
BVC, LBVC	Branch if overflow clear
	SIGNED BRANCHES
BGT, LBGT	Branch if greater (signed)
BGE, LBGE	Branch if greater than or equal (signed)
BEQ, LBEQ	Branch if equal
BLE, LBLE	Branch if less than or equal (signed)
BLT, LBLT	Branch if less than (signed)
	UNSIGNED BRANCHES
BHI, LBHI	Branch if higher (unsigned)
BHS, LBHS	Branch if higher or same (unsigned)
BEQ, LBEQ	Branch if equal
BLS, LBLS	Branch if lower or same (unsigned)
BLO, LBLO	Branch if lower (unsigned)
	OTHER BRANCHES
BSR, LBSR	Branch to subroutine
BRA, LBRA	Branch always
BRN, LBRN	Branch never

Table 8 Miscellaneous Instructions

Mnemonic(s)	Operation
ANDCC	AND condition code register
CWAI	AND condition code register, then wait for interrupt
NOP	No operation
ORCC	OR condition code register
JMP	Jump
JSR	Jump to subroutine
RTI	Return from interrupt
RTS	Return from subroutine
SWI, SWI2, SWI3	Software interrupt (absolute indirect)
SYNC	Synchronize with interrupt line

Table 9 HD6809E Instruction Set Table

INSTRUCTION/ FORMS		HD6809E ADDRESSING MODES																	DESCRIPTION	5	3	2	1	0
		IMPLIED			DIRECT			EXTENDED			IMMEDIATE			INDEXED①			RELATIVE				H	N	Z	V	C
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~⑤	#						
ABX		3A	3	1																B + X → X (UNSIGNED)	•	•	•	•	•
ADC	ADCA				99	4	2	B9	5	3	89	2	2	A9	4+	2+				A + M + C → A	↕	↕	↕	↕	↕
	ADCB				D9	4	2	F9	5	3	C9	2	2	E9	4+	2+				B + M + C → B	↕	↕	↕	↕	↕
ADD	ADDA				9B	4	2	BB	5	3	8B	2	2	AB	4+	2+				A + M → A	↕	↕	↕	↕	↕
	ADDB				DB	4	2	FB	5	3	CB	2	2	EB	4+	2+				B + M → B	↕	↕	↕	↕	↕
	ADDD				D3	6	2	F3	7	3	C3	4	3	E3	6+	2+				D + M:M + 1 → D	•	↕	↕	↕	↕
AND	ANDA				94	4	2	B4	5	3	84	2	2	A4	4+	2+				A ∧ M → A	•	↕	↕	0	•
	ANDB				D4	4	2	F4	5	3	C4	2	2	E4	4+	2+				B ∧ M → B	•	↕	↕	0	•
	ANDCC										1C	3	2							CC ∧ IMM → CC	(			(7)	)
ASL	ASLA	48	2	1																A	⑧	↕	↕	↕	↕
	ASLB	58	2	1																B	⑧	↕	↕	↕	↕
	ASL				08	6	2	78	7	3				68	6+	2+				M	⑧	↕	↕	↕	↕
ASR	ASRA	47	2	1																A	⑧	↕	↕	•	↕
	ASRB	57	2	1																B	⑧	↕	↕	•	↕
	ASR				07	6	2	77	7	3				67	6+	2+				M	⑧	↕	↕	•	↕
BCC	BCC																24	3	2	Branch C = 0	•	•	•	•	•
	LBCC																10 24	5(6)	4	Long Branch C = 0	•	•	•	•	•
BCS	BCS																25	3	2	Branch C = 1	•	•	•	•	•
	LBCS																10 25	5(6)	4	Long Branch C = 1	•	•	•	•	•
BEQ	BEQ																27	3	2	Branch Z = 1	•	•	•	•	•
	LBEQ																10 27	5(6)	4	Long Branch Z = 1	•	•	•	•	•
BGE	BGE																2C	3	2	Branch N ⊕ V=0	•	•	•	•	•
	LBGE																10 2C	5(6)	4	Long Branch N ⊕ V=0	•	•	•	•	•
BGT	BGT																2E	3	2	Branch Z∨(N⊕V)=0	•	•	•	•	•
	LBGT																10 2E	5(6)	4	Long Branch Z∨(N ⊕ V)=0	•	•	•	•	•
BHI	BHI																22	3	2	Branch C∨Z=0	•	•	•	•	•
	LBHI																10 22	5(6)	4	Long Branch C∨Z=0	•	•	•	•	•
BHS	BHS																24	3	2	Branch C=0	•	•	•	•	•
	LBHS																10 24	5(6)	4	Long Branch C=0	•	•	•	•	•
BIT	BITA				95	4	2	B5	5	3	85	2	2	A5	4+	2+				Bit Test A (M ∧ A)	•	↕	↕	0	•
	BITB				D5	4	2	F5	5	3	C5	2	2	E5	4+	2+				Bit Test B (M ∧ B)	•	↕	↕	0	•
BLE	BLE																2F	3	2	Branch Z∨(N⊕V)=1	•	•	•	•	•
	LBLE																10 2F	5(6)	4	Long Branch Z∨(N ⊕ V)=1	•	•	•	•	•
BLO	BLO																25	3	2	Branch C=1	•	•	•	•	•
	LBLO																10 25	5(6)	4	Long Branch C=1	•	•	•	•	•
BLS	BLS																23	3	2	Branch C∨Z=1	•	•	•	•	•
	LBLS																10 23	5(6)	4	Long Branch C∨Z=1	•	•	•	•	•
BLT	BLT																2D	3	2	Branch N ⊕ V=1	•	•	•	•	•
	LBLT																10 2D	5(6)	4	Long Branch N ⊕ V=1	•	•	•	•	•
BMI	BMI																2B	3	2	Branch N=1	•	•	•	•	•
	LBMI																10 2B	5(6)	4	Long Branch N=1	•	•	•	•	•
BNE	BNE																26	3	2	Branch Z = 0	•	•	•	•	•
	LBNE																10 26	5(6)	4	Long Branch Z = 0	•	•	•	•	•
BPL	BPL																2A	3	2	Branch N = 0	•	•	•	•	•
	LBPL																10 2A	5(6)	4	Long Branch N = 0	•	•	•	•	•
BRA	BRA																20	3	2	Branch Always	•	•	•	•	•
	LBRA																16	5	3	Long Branch/ Always	•	•	•	•	•
BRN	BRN																21	3	2	Branch Never	•	•	•	•	•
	LBRN																10 21	5	4	Long Branch Never	•	•	•	•	•

(to be continued)

INSTRUCTION/ FORMS		HD6809E ADDRESSING MODES																	DESCRIPTION	5	3	2	1	0
		IMPLIED			DIRECT			EXTENDED			IMMEDIATE			INDEXED①			RELATIVE				H	N	Z	V	C
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~⑤	#						
BSR	BSR																8D	7	2	Branch to Subroutine	•	•	•	•	•
	LBSR																17	9	3	Long Branch to Subroutine	•	•	•	•	•
BVC	BVC																28	3	2	Branch V = 0	•	•	•	•	•
	LBVC																10 28	5(6)	4	Long Branch V = 0	•	•	•	•	•
BVS	BVS																29	3	2	Branch V = 1	•	•	•	•	•
	LBVS																10 29	5(6)	4	Long Branch V = 1	•	•	•	•	•
CLR	CLRA	4F	2	1																0 → A	•	0	1	0	0
	CLRB	5F	2	1																0 → B	•	0	1	0	0
	CLR				0F	6	2	7F	7	3				6F	6+	2+				0 → M	•	0	1	0	0
CMP	CMPA				91	4	2	B1	5	3	81	2	2	A1	4+	2+				Compare M from A	⑧	↕	↕	↕	↕
	CMPB				D1	4	2	F1	5	3	C1	2	2	E1	4+	2+				Compare M from B	⑧	↕	↕	↕	↕
	CMPD				10 93	7	3	10 B3	8	4	10 83	5	4	10 A3	7+	3+				Compare M: M + 1 from D	•	↕	↕	↕	↕
	CMPS				11 9C	7	3	11 BC	8	4	11 8C	5	4	11 AC	7+	3+				Compare M: M + 1 from S	•	↕	↕	↕	↕
	CMPU				11 93	7	3	11 B3	8	4	11 83	5	4	11 A3	7+	3+				Compare M: M + 1 from U	•	↕	↕	↕	↕
	CMPX				9C	6	2	BC	7	3	8C	4	3	AC	6+	2+				Compare M: M + 1 from X	•	↕	↕	↕	↕
	CMPY				10 9C	7	3	10 BC	8	4	10 8C	5	4	10 AC	7+	3+				Compare M: M + 1 from Y	•	↕	↕	↕	↕
COM	COMA	43	2	1																$\overline{A} \rightarrow A$	•	↕	↕	0	1
	COMB	53	2	1																$\overline{B} \rightarrow B$	•	↕	↕	0	1
	COM				03	6	2	73	7	3				63	6+	2+				$\overline{M} \rightarrow M$	•	↕	↕	0	1
CWAI		3C	20	2																CC ∧ IMM → CC (except 1→E) Wait for Interrupt	(———	⑦	———	———	)
DAA		19	2	1																Decimal Adjust A	•	↕	↕	⑧	↕
DEC	DECA	4A	2	1																A − 1 → A	•	↕	↕	↕	•
	DECB	5A	2	1																B − 1 → B	•	↕	↕	↕	•
	DEC				0A	6	2	7A	7	3				6A	6+	2+				M − 1 → M	•	↕	↕	↕	•
EOR	EORA				98	4	2	B8	5	3	88	2	2	A8	4+	2+				A ⊕ M → A	•	↕	↕	0	•
	EORB				D8	4	2	F8	5	3	C8	2	2	E8	4+	2+				B ⊕ M → B	•	↕	↕	0	•
EXG	R1, R2	1E	7	2																R1 ↔ R2②	(———	⑩	———	———	)
INC	INCA	4C	2	1																A + 1 → A	•	↕	↕	↕	•
	INCB	5C	2	1																B + 1 → B	•	↕	↕	↕	•
	INC				0C	6	2	7C	7	3				6C	6+	2+				M + 1 → M	•	↕	↕	↕	•
JMP					0E	3	2	7E	4	3				6E	3+	2+				EA② → PC	•	•	•	•	•
JSR					9D	7	2	BD	8	3				AD	7+	2+				Jump to Subroutine	•	•	•	•	•
LD	LDA				96	4	2	B6	5	3	86	2	2	A6	4+	2+				M → A	•	↕	↕	0	•
	LDB				D6	4	2	F6	5	3	C6	2	2	E6	4+	2+				M → B	•	↕	↕	0	•
	LDD				DC	5	2	FC	6	3	CC	3	3	EC	5+	2+				M: M + 1 → D	•	↕	↕	0	•
	LDS				10 DE	6	3	10 FE	7	4	10 CE	4	4	10 EE	6+	3+				M: M + 1 → S	•	↕	↕	0	•
	LDU				DE	5	2	FE	6	3	CE	3	3	EE	5+	2+				M: M + 1 → U	•	↕	↕	0	•
	LDX				9E	5	2	BE	6	3	8E	3	3	AE	5+	2+				M: M + 1 → X	•	↕	↕	0	•
	LDY				10 9E	6	3	10 BE	7	4	10 8E	4	4	10 AE	6+	3+				M: M + 1 → Y	•	↕	↕	0	•
LEA	LEAS													32	4+	2+				EA② → S	•	•	•	•	•
	LEAU													33	4+	2+				EA② → U	•	•	•	•	•
	LEAX													30	4+	2+				EA② → X	•	•	↕	•	•
	LEAY													31	4+	2+				EA② → Y	•	•	↕	•	•
LSL	LSLA	48	2	1																A B M	•	↕	↕	↕	↕
	LSLB	58	2	1																	•	↕	↕	↕	↕
	LSL				08	6	2	78	7	3				68	6+	2+					•	↕	↕	↕	↕
LSR	LSRA	44	2	1																A B M	•	0	↕	•	↕
	LSRB	54	2	1																	•	0	↕	•	↕
	LSR				04	6	2	74	7	3				64	6+	2+					•	0	↕	•	↕
MUL		3D	11	1																A × B → D (Unsigned)	•	•	↕	•	⑨
NEG	NEGA	40	2	1																$\overline{A} + 1 \rightarrow A$	⑧	↕	↕	↕	↕
	NEGB	50	2	1																$\overline{B} + 1 \rightarrow B$	⑧	↕	↕	↕	↕
	NEG				00	6	2	70	7	3				60	6+	2+				$\overline{M} + 1 \rightarrow M$	⑧	↕	↕	↕	↕
NOP		12	2	1																No Operation	•	•	•	•	•

(to be continued)

INSTRUCTION/ FORMS		HD6809E ADDRESSING MODES																	DESCRIPTION	5	3	2	1	0
		IMPLIED			DIRECT			EXTENDED			IMMEDIATE			INDEXED①			RELATIVE				H	N	Z	V	C
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~⑤	#						
OR	ORA				9A	4	2	BA	5	3	8A	2	2	AA	4+	2+				A ∨ M → A	•	↕	↕	0	•
	ORB				DA	4	2	FA	5	3	CA	2	2	EA	4+	2+				B ∨ M → B	•	↕	↕	0	•
	ORCC										1A	3	2							CC ∨ IMM → CC	(——		⑦		——)
PSH	PSHS	34	5+④	2																Push Registers on S Stack	•	•	•	•	•
	PSHU	36	5+④	2																Push Registers on U Stack	•	•	•	•	•
PUL	PULS	35	5+④	2																Pull Registers from S Stack	(——		⑩		——)
	PULU	37	5+④	2																Pull Registers from U Stack	(——		⑩		——)
ROL	ROLA	49	2	1																A }	•	↕	↕	↕	↕
	ROLB	59	2	1																B }	•	↕	↕	↕	↕
	ROL				09	6	2	79	7	3				69	6+	2+				M }	•	↕	↕	↕	↕
ROR	RORA	46	2	1																A }	•	↕	↕	•	↕
	RORB	56	2	1																B }	•	↕	↕	•	↕
	ROR				06	6	2	76	7	3				66	6+	2+				M }	•	↕	↕	•	↕
RTI		3B	6/15	1																Return From Interrupt	(——		⑦		——)
RTS		39	5	1																Return From Subroutine	•	•	•	•	•
SBC	SBCA				92	4	2	B2	5	3	82	2	2	A2	4+	2+				A − M − C → A	⑧	↕	↕	↕	↕
	SBCB				D2	4	2	F2	5	3	C2	2	2	E2	4+	2+				B − M − C → B	⑧	↕	↕	↕	↕
SEX		1D	2	1																Sign Extend B into A	•	↕	↕	•	•
ST	STA				97	4	2	B7	5	3				A7	4+	2+				A → M	•	↕	↕	0	•
	STB				D7	4	2	F7	5	3				E7	4+	2+				B → M	•	↕	↕	0	•
	STD				DD	5	2	FD	6	3				ED	5+	2+				D → M: M+1	•	↕	↕	0	•
	STS				10 DF	6	3	10 FF	7	4				10 EF	6+	3+				S → M: M + 1	•	↕	↕	0	•
	STU				DF	5	2	FF	6	3				EF	5+	2+				U → M: M + 1	•	↕	↕	0	•
	STX				9F	5	2	BF	6	3				AF	5+	2+				X → M: M + 1	•	↕	↕	0	•
	STY				10 9F	6	3	10 BF	7	4				10 AF	6+	3+				Y → M: M + 1	•	↕	↕	0	•
SUB	SUBA				90	4	2	B0	5	3	80	2	2	A0	4+	2+				A − M → A	⑧	↕	↕	↕	↕
	SUBB				D0	4	2	F0	5	3	C0	2	2	E0	4+	2+				B − M → B	⑧	↕	↕	↕	↕
	SUBD				93	6	2	B3	7	3	83	4	3	A3	6+	2+				D − M: M + 1 → D	•	↕	↕	↕	↕
SWI	SWI⑥	3F	19	1																Software Interrupt1	•	•	•	•	•
	SWI2⑥	10 3F	20	2																Software Interrupt2	•	•	•	•	•
	SWI3⑥	11 3F	20	2																Software Interrupt3	•	•	•	•	•
SYNC		13	≥2	1																Synchronize to Interrupt	•	•	•	•	•
TFR	R1, R2	1F	6	2																R1 → R2⑦	(——		⑩		——)
TST	TSTA	4D	2	1																Test A	•	↕	↕	0	•
	TSTB	5D	2	1																Test B	•	↕	↕	0	•
	TST				0D	6	2	7D	7	3				6D	6+	2+				Test M	•	↕	↕	0	•

(NOTES)

① This column gives a base cycle and byte count. To obtain total count, and the values obtained from the INDEXED ADDRESSING MODES table.
② R1 and R2 may be any pair of 8 bit or any pair of 16 bit registers.
The 8 bit registers are: A, B, CC, DP
The 16 bit registers are: X, Y, U, S, D, PC
③ EA is the effective address.
④ The PSH and PUL instructions require 5 cycle plus 1 cycle for each byte pushed or pulled.
⑤ 5(6) means: 5 cycles if branch not taken, 6 cycles if taken.
⑥ SWI sets I and F bits. SWI2 and SWI3 do not affect I and F.
⑦ Conditions Codes set as a direct result of the instruction.
⑧ Value of half-carry flag is undefined.
⑨ Special Case — Carry set if b7 is SET.
⑩ Condition Codes set as a direct result of the instruction if CC is specified, and not affected otherwise.

LEGEND:

OP	Operation Code (Hexadecimal)	Z	Zero (byte)
~	Number of MPU Cycles	V	Overflow, 2's complement
#	Number of Program Bytes	C	Carry from bit 7
+	Arithmetic Plus	↕	Test and set if true, cleared otherwise
−	Arithmetic Minus	•	Not Affected
×	Multiply	CC	Condition Code Register
M̄	Complement of M	:	Concatenation
→	Transfer Into	∨	Logical or
H	Half-carry (from bit 3)	∧	Logical and
N	Negative (sign bit)	⊕	Logical Exclusive or

Table 10 Hexadecimal Values of Machine Codes

OP	Mnem	Mode	~	#	OP	Mnem	Mode	~	#	OP	Mnem	Mode	~	#
00	NEG	Direct	6	2	30	LEAX	Indexed	4+	2+	60	NEG	Indexed	6+	2+
01	*	↑			31	LEAY	↑	4+	2+	61	*	↑		
02	*				32	LEAS	↓	4+	2+	62	*			
03	COM		6	2	33	LEAU	Indexed	4+	2+	63	COM		6+	2+
04	LSR		6	2	34	PSHS	Implied	5+	2	64	LSR		6+	2+
05	*				35	PULS	↑	5+	2	65	*			
06	ROR		6	2	36	PSHU		5+	2	66	ROR		6+	2+
07	ASR		6	2	37	PULU		5+	2	67	ASR		6+	2+
08	ASL, LSL		6	2	38	*				68	ASL, LSL		6+	2+
09	ROL		6	2	39	RTS		5	1	69	ROL		6+	2+
0A	DEC		6	2	3A	ABX		3	1	6A	DEC		6+	2+
0B	*				3B	RTI		6, 15	1	6B	*			
0C	INC		6	2	3C	CWAI		20	2	6C	INC		6+	2+
0D	TST		6	2	3D	MUL		11	1	6D	TST		6+	2+
0E	JMP		3	2	3E	*				6E	JMP		3+	2+
0F	CLR	Direct	6	2	3F	SWI	Implied	19	1	6F	CLR	Indexed	6+	2+
10	See	—	—	—	40	NEGA	Implied	2	1	70	NEG	Extended	7	3
11	Next Page	—	—	—	41	*	↑			71	*	↑		
12	NOP	Implied	2	1	42	*				72	*			
13	SYNC	Implied	2	1	43	COMA		2	1	73	COM		7	3
14	*				44	LSRA		2	1	74	LSR		7	3
15	*				45	*				75	*			
16	LBRA	Relative	5	3	46	RORA		2	1	76	ROR		7	3
17	LBSR	Relative	9	3	47	ASRA		2	1	77	ASR		7	3
18	*				48	ASLA, LSLA		2	1	78	ASL, LSL		7	3
19	DAA	Implied	2	1	49	ROLA		2	1	79	ROL		7	3
1A	ORCC	Immed	3	2	4A	DECA		2	1	7A	DEC		7	3
1B	*	—			4B	*				7B	*			
1C	ANDCC	Immed	3	2	4C	INCA		2	1	7C	INC		7	3
1D	SEX	Implied	2	1	4D	TSTA		2	1	7D	TST		7	3
1E	EXG	↕	8	2	4E	*				7E	JMP		4	3
1F	TFR	Implied	6	2	4F	CLRA	Implied	2	1	7F	CLR	Extended	7	3
20	BRA	Relative	3	2	50	NEGB	Implied	2	1	80	SUBA	Immed	2	2
21	BRN	↑	3	2	51	*	↑			81	CMPA	↑	2	2
22	BHI		3	2	52	*				82	SBCA		2	2
23	BLS		3	2	53	COMB		2	1	83	SUBD		4	3
24	BHS, BCC		3	2	54	LSRB		2	1	84	ANDA		2	2
25	BLO, BCS		3	2	55	*				85	BITA		2	2
26	BNE		3	2	56	RORB		2	1	86	LDA		2	2
27	BEQ		3	2	57	ASRB		2	1	87	*			
28	BVC		3	2	58	ASLB, LSLB		2	1	88	EORA		2	2
29	BVS		3	2	59	ROLB		2	1	89	ADCA		2	2
2A	BPL		3	2	5A	DECB		2	1	8A	ORA		2	2
2B	BMI		3	2	5B	*				8B	ADDA		2	2
2C	BGE		3	2	5C	INCB		2	1	8C	CMPX	Immed	4	3
2D	BLT		3	2	5D	TSTB		2	1	8D	BSR	Relative	7	2
2E	BGT		3	2	5E	*				8E	LDX	Immed	3	3
2F	BLE	Relative	3	2	5F	CLRB	Implied	2	1	8F	*			

LEGEND:
~ Number of MPU cycles (less possible push pull or indexed-mode cycles)
# Number of program bytes
* Denotes unused opcode

(to be continued)

OP	Mnem	Mode	~	#
90	SUBA	Direct	4	2
91	CMPA	↑	4	2
92	SBCA		4	2
93	SUBD		6	2
94	ANDA		4	2
95	BITA		4	2
96	LDA		4	2
97	STA		4	2
98	EORA		4	2
99	ADCA		4	2
9A	ORA		4	2
9B	ADDA		4	2
9C	CMPX		6	2
9D	JSR		7	2
9E	LDX		5	2
9F	STX	↓ Direct	5	2
A0	SUBA	Indexed	4+	2+
A1	CMPA	↑	4+	2+
A2	SBCA		4+	2+
A3	SUBD		6+	2+
A4	ANDA		4+	2+
A5	BITA		4+	2+
A6	LDA		4+	2+
A7	STA		4+	2+
A8	EORA		4+	2+
A9	ADCA		4+	2+
AA	ORA		4+	2+
AB	ADDA		4+	2+
AC	CMPX		6+	2+
AD	JSR		7+	2+
AE	LDX		5+	2+
AF	STX	↓ Indexed	5+	2+
B0	SUBA	Extended	5	3
B1	CMPA	↑	5	3
B2	SBCA		5	3
B3	SUBD		7	3
B4	ANDA		5	3
B5	BITA		5	3
B6	LDA		5	3
B7	STA		5	3
B8	EORA		5	3
B9	ADCA		5	3
BA	ORA		5	3
BB	ADDA		5	3
BC	CMPX		7	3
BD	JSR		8	3
BE	LDX		6	3
BF	STX	↓ Extended	6	3
C0	SUBB	Immed	2	2
C1	CMPB	↑	2	2
C2	SBCB		2	2
C3	ADDD		4	3
C4	ANDB		2	2
C5	BITB	↓ Immed	2	2
C6	LDB	Immed	2	2
C7	*	↑		
C8	EORB		2	2
C9	ADCB		2	2
CA	ORB		2	2
CB	ADDB		2	2
CC	LDD		3	3
CD	*			
CE	LDU	↓ Immed	3	3
CF	*			
D0	SUBB	Direct	4	2
D1	CMPB	↑	4	2
D2	SBCB		4	2
D3	ADDD		6	2
D4	ANDB		4	2
D5	BITB		4	2
D6	LDB		4	2
D7	STB		4	2
D8	EORB		4	2
D9	ADCB		4	2
DA	ORB		4	2
DB	ADDB		4	2
DC	LDD		5	2
DD	STD		5	2
DE	LDU		5	2
DF	STU	↓ Direct	5	2
E0	SUBB	Indexed	4+	2+
E1	CMPB	↑	4+	2+
E2	SBCB		4+	2+
E3	ADDD		6+	2+
E4	ANDB		4+	2+
E5	BITB		4+	2+
E6	LDB		4+	2+
E7	STB		4+	2+
E8	EORB		4+	2+
E9	ADCB		4+	2+
EA	ORB		4+	2+
EB	ADDB		4+	2+
EC	LDD		5+	2+
ED	STD		5+	2+
EE	LDU		5+	2+
EF	STU	↓ Indexed	5+	2+
F0	SUBB	Extended	5	3
F1	CMPB	↑	5	3
F2	SBCB		5	3
F3	ADDD		7	3
F4	ANDB		5	3
F5	BITB		5	3
F6	LDB		5	3
F7	STB		5	3
F8	EORB		5	3
F9	ADCB		5	3
FA	ORB		5	3
FB	ADDB	↓ Extended	5	3
FC	LDD	Extended	6	3
FD	STD	↑	6	3
FE	LDU		6	3
FF	STU	↓ Extended	6	3

2 Bytes Opcode

1021	LBRN	Relative	5	4
1022	LBHI	↑	5(6)	4
1023	LBLS		5(6)	4
1024	LBHS, LBCC		5(6)	4
1025	LBCS, LBLO		5(6)	4
1026	LBNE		5(6)	4
1027	LBEQ		5(6)	4
1028	LBVC		5(6)	4
1029	LBVS		5(6)	4
102A	LBPL		5(6)	4
102B	LBMI		5(6)	4
102C	LBGE		5(6)	4
102D	LBLT		5(6)	4
102E	LBGT		5(6)	4
102F	LBLE	↓ Relative	5(6)	4
103F	SWI2	Implied	20	2
1083	CMPD	Immed	5	4
108C	CMPY	↑↓	5	4
108E	LDY	Immed	4	4
1093	CMPD	Direct	7	3
109C	CMPY	↑	7	3
109E	LDY	↓	6	3
109F	STY	Direct	6	3
10A3	CMPD	Indexed	7+	3+
10AC	CMPY	↑	7+	3+
10AE	LDY	↓	6+	3+
10AF	STY	Indexed	6+	3+
10B3	CMPD	Extended	8	4
10BC	CMPY	↑	8	4
10BE	LDY	↓	7	4
10BF	STY	Extended	7	4
10CE	LDS	Immed	4	4
10DE	LDS	Direct	6	3
10DF	STS	Direct	6	3
10EE	LDS	Indexed	6+	3+
10EF	STS	Indexed	6+	3+
10FE	LDS	Extended	7	4
10FF	STS	Extended	7	4
113F	SWI3	Implied	20	2
1183	CMPU	Immed	5	4
118C	CMPS	Immed	5	4
1193	CMPU	Direct	7	3
119C	CMPS	Direct	7	3
11A3	CMPU	Indexed	7+	3+
11AC	CMPS	Indexed	7+	3+
11B3	CMPU	Extended	8	4
11BC	CMPS	Extended	8	4

(NOTE): All unused opcodes are both undefined and illegal

■ NOTE FOR USE

Execution Sequence of CLR Instruction

Cycle-by-cycle flow of CLR instruction (Direct, Extended, Indexed Addressing Mode) is shown below. In this sequence the content of the memory location specified by the operand is read before writing "00" into it. Note that status Flags, such as IRQ Flag, will be cleared by this extra data read operation when accessing the control/status register (sharing the same address between read and write) of peripheral devices.

Example: CLR (Extended)

$8000	CLR	$A000
$A000	FCB	$80

Cycle #	Address	Data	R/$\overline{W}$	Description
1	8000	7F	1	Opcode Fetch
2	8001	A0	1	Operand Address, High Byte
3	8002	00	1	Operand Address, Low Byte
4	FFFF	*	1	$\overline{VMA}$ Cycle
5	A000	80	1	Read the Data
6	FFFF	*	1	$\overline{VMA}$ Cycle
7	A000	00	0	Store Fixed "00" into Specified Location

* The data bus has the data at that particular address.

HD6309E
CMOS MPU (Micro Processing Unit)

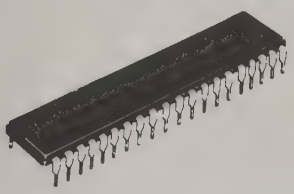
—PRELIMINARY—

The HD6309E is the highest 8-bit microprocessor of HMCS6800 family, which is just compatible with the conventional HD6809E.

The HD6309E has hardware and software features which make it an ideal processor for higher level language execution or standard controller applications. External clock inputs are provided to allow synchronization with peripherals, systems or other MPUs.

The HD6309E is complete CMOS device and the power dissipation is extremely low. Moreover, the SYNC and CWAI instruction makes low power application possible.

HD6309EP



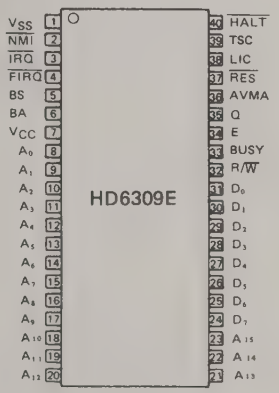
(DP-40)

■ FEATURES

- Hardware — Interface with All HMCS6800 Peripherals
- Software — Object Code Compatible with the HD6809E
- Low Power Consumption Mode (Sleep mode)
  - SYNC state of SYNC Instruction
  - WAIT state of CWAI Instruction
- External Clock Inputs, E and Q, Allow Synchronization
- Wide Operation Range
  - f = 0.5 to 3MHz (V_{CC}=5V±10%)

Bus Timing
2.0MHz
2.5MHz
3.0MHz

■ PIN ARRANGEMENT



(Top View)

■ ABSOLUTE MAXIMUM RATINGS

Item	Symbol	Value	Unit
Supply Voltage	V_{CC}*	-0.3 ~ +7.0	V
Input Voltage	V_{in}*	-0.3 ~ +7.0	V
Maximum Output Current	$\|I_O\|$**	5	mA
Maximum Total Output Current	$\|\Sigma I_O\|$***	100	mA
Operating Temperature	T_{opr}	-20 ~ +75	°C
Storage Temperature	T_{stg}	-55 ~ +150	°C

* With respect to V_{SS} (SYSTEM GND)
** Maximum output current is the maximum currents which can flow out from one output terminal and I/O common terminal. ($A_0 \sim A_{15}$, R/$\overline{W}$, $D_0 \sim D_7$, BA, BS, LIC, AVMA, BUSY)
*** Maximum total output current is the total sum of output currents which can flow out simultaneously from output terminals and I/O common terminals. ($A_0 \sim A_{15}$, R/$\overline{W}$, $D_0 \sim D_7$, BA, BS, LIC, AVMA, BUSY)
(NOTE) Permanent LSI damage may occur if maximum ratings are exceeded. Normal operation should be under recommended operating conditions. If these conditions are exceeded, it could affect reliability of LSI.

■ RECOMMENDED OPERATING CONDITIONS

Item		Symbol	min	typ	max	Unit
Supply Voltage		V_{CC}*	4.5	5.0	5.5	V
Input Voltage	Logic, Q, $\overline{RES}$	V_{IL}*	-0.3	–	0.8	V
	E	V_{ILC}*	-0.3	–	0.4	V
	Logic, Q, E	V_{IH}*	2.0	–	V_{CC}	V
	$\overline{RES}$		V_{CC}-0.5	–	V_{CC}	V
Operating Temperature		T_{opr}	-20	25	75	°C

* With respect to V_{SS} (SYSTEM GND)

■ ELECTRICAL CHARACTERISTICS

● DC CHARACTERISTICS (V_{CC}=5.0V±10%, V_{SS}=0V, Ta=- 20 ~ +75°C, unless otherwise noted.)

Item		Symbol	Test Condition	2.0MHz			2.5MHz			3.0MHz			Unit
				min	typ*	max	min	typ*	max	min	typ*	max	
Input "High" Voltage	Logic, Q, E	V_{IH}		2.0	–	V_{CC}	2.0	–	V_{CC}	2.0	–	V_{CC}	V
	$\overline{RES}$	V_{IHR}		V_{CC}-0.5	–	V_{CC}	V_{CC}-0.5	–	V_{CC}	V_{CC}-0.5	–	V_{CC}	V
Input "Low" Voltage	Logic, Q, $\overline{RES}$	V_{IL}		-0.3	–	0.8	-0.3	–	0.8	-0.3	–	0.8	V
	E	V_{ILC}		-0.3	–	0.4	-0.3	–	0.4	-0.3	–	0.4	V
Input Leakage Current	Logic, Q, $\overline{RES}$	I_{in}	V_{in}=0 ~ V_{CC}, V_{CC}=max	-2.5	–	2.5	-2.5	–	2.5	-2.5	–	2.5	μA
	E			-10	–	10	-10	–	10	-10	–	10	μA
Output "High" Voltage	$D_0 \sim D_7$	V_{OH}	I_{LOAD}=-400μA	4.1	–	–	4.1	–	–	4.1	–	–	V
			$I_{LOAD} \leqq$ -10μA	V_{CC}-0.1	–	–	V_{CC}-0.1	–	–	V_{CC}-0.1	–	–	
	$A_0 \sim A_{15}$, R/$\overline{W}$		I_{LOAD} = -400μA	4.1	–	–	4.1	–	–	4.1	–	–	V
			$I_{LOAD} \leqq$ -10μA	V_{CC}-0.1	–	–	V_{CC}-0.1	–	–	V_{CC}-0.1	–	–	
	BA, BS, LIC, AVMA, BUSY		I_{LOAD}=-400μA	4.1	–	–	4.1	–	–	4.1	–	–	V
			$I_{LOAD} \leqq$ -10μA	V_{CC}-0.1	–	–	V_{CC}-0.1	–	–	V_{CC}-0.1	–	–	
Output "Low" Voltage		V_{OL}	I_{LOAD}=2mA	–	–	0.5	–	–	0.5	–	–	0.5	V
Input Capacitance	$D_0 \sim D_7$, Logic Input, Q, $\overline{RES}$	C_{in}	V_{in}=0V, Ta=25°C, f=1MHz	–	10	15	–	10	15	–	10	15	pF
	E			–	30	50	–	30	50	–	30	50	pF
Output Capacitance	$A_0 \sim A_{15}$, R/$\overline{W}$, BA, BS, LIC, AVMA, BUSY	C_{out}	V_{in}=0V, Ta=25°C, f=1MHz	–	10	15	–	10	15	–	10	15	pF
Frequency of Operation	E, Q	f		0.5	–	2.0	0.5	–	2.5	0.5	–	3.0	MHz
Three-State (Off State) Input Current	$D_0 \sim D_7$	I_{TSI}	V_{in}=0.4~V_{CC}, V_{CC}=max	-10	–	10	-10	–	10	-10	–	10	μA
	$A_0 \sim A_{15}$, R/$\overline{W}$			-10	–	10	-10	–	10	-10	–	10	μA
Current Dissipation		I_{CC}	Operating	–	–	12	–	–	15	–	–	18	mA
			Sleeping	–	–	2	–	–	2	–	–	2	

*Ta=25°C, V_{CC}=5V

- **AC CHARACTERISTICS (V_{CC}=5.0V±10%, V_{SS}=0, Ta=−20 ~ +75°C, unless otherwise noted.)**

1. CLOCK TIMING

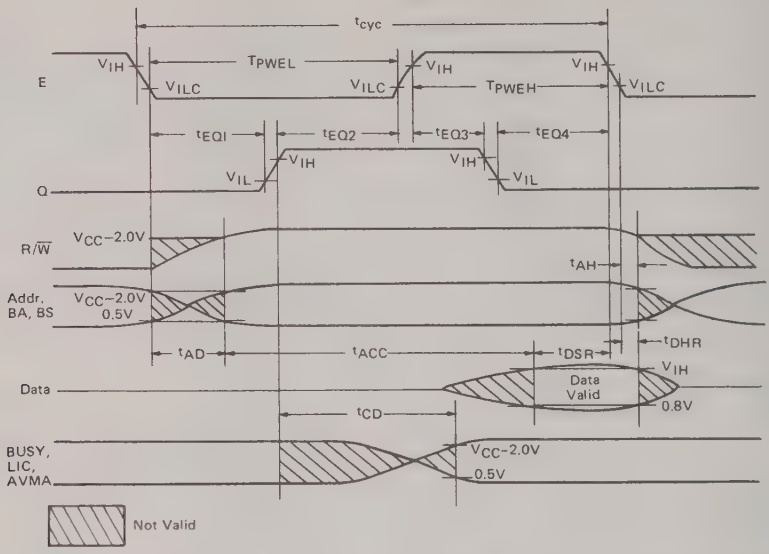
Item		Symbol	Test Condition	2.0MHz			2.5MHz			3.0MHz			Unit
				min	typ	max	min	typ	max	min	typ	max	
Cycle Time		t_{cyc}	Fig. 1, 2	500	–	2000	400	–	2000	333	–	2000	ns
E Clock "Low"		t_{PWEL}		210	–	1000	170	–	1000	140	–	1000	ns
E Clock "High" (Measured at V_{IH})		t_{PWEH}		220	–	1000	170	–	1000	140	–	1000	ns
E Rise and Fall Time		t_{Er}, t_{Ef}		–	–	20	–	–	15	–	–	15	ns
Q Clock "High"		t_{PWQH}		220	–	1000	170	–	1000	140	–	1000	ns
Q Rise and Fall Time		t_{Qr}, t_{Qf}		–	–	20	–	–	15	–	–	15	ns
E "Low" to Q Rising	E "Low"→Q "High"	t_{EQ1}		100	–	–	80	–	–	65	–	–	ns
Q "High" to E Rising	Q "High"→E "High"	t_{EQ2}		100	–	–	80	–	–	65	–	–	ns
E "High" to Q Falling	E "High"→Q "Low"	t_{EQ3}		100	–	–	80	–	–	65	–	–	ns
Q "Low" to E Falling	Q "Low"→E "Low"	t_{EQ4}		100	–	–	80	–	–	65	–	–	ns

2. BUS TIMING

Item	Symbol	Test Condition	2.0MHz			2.5MHz			3.0MHz			Unit
			min	typ	max	min	typ	max	min	typ	max	
Address Delay	t_{AD}	Fig. 1, 2	–	–	110	–	–	85	–	–	70	ns
Address Hold Time (Address, R/$\overline{W}$, BA, BS)	t_{AH}		20	–	–	20	–	–	20	–	–	ns
Peripheral Read Access Times ($t_{cyc}-t_{Ef}-t_{AD}-t_{DSR}=t_{ACC}$)	t_{ACC}		330	–	–	260	–	–	205	–	–	ns
Data Setup Time (Read)	t_{DSR}		40	–	–	40	–	–	40	–	–	ns
Input Data Hold Time	t_{DHR}		10	–	–	10	–	–	10	–	–	ns
Data Delay Time (Write)	t_{DDW}		–	–	110	–	–	85	–	–	70	ns
Output Data Hold Time	t_{DHW}		30	–	70	30	–	70	30	–	70	ns

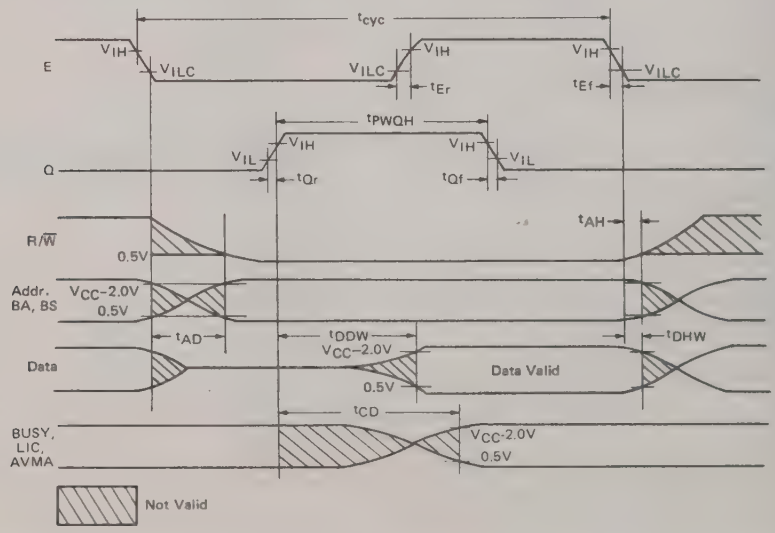
3. PROCESSOR CONTROL TIMING

Item	Symbol	Test Condition	2.0MHz			2.5MHz			3.0MHz			Unit
			min	typ	max	min	typ	max	min	typ	max	
Control Delay (BUSY, LIC, AVMA)	t_{CD}	Fig. 1, 2, 7 ~ 10, 14 and 17	–	–	200	–	–	160	–	–	130	ns
Interrupts Set Up Time	t_{PCS}		110	–	–	90	–	–	70	–	–	ns
$\overline{HALT}$ Set Up Time	t_{PCS}		110	–	–	90	–	–	70	–	–	ns
$\overline{RES}$ Set UpTime	t_{PCS}		110	–	–	90	–	–	70	–	–	ns
TSC Setup Time	t_{PCS}		110	–	–	90	–	–	70	–	–	ns
TSC Drive to Valid Logic Levels	t_{TSA}		–	–	120	–	–	95	–	–	80	ns
TSC Release MOS Buffers to High Impedance	t_{TSR}		–	–	110	–	–	85	–	–	70	ns
TSC Three-State Delay	t_{TSD}		–	–	80	–	–	60	–	–	50	ns
Processor Control Rise/Fall	t_{PCr}, t_{PCf}		–	–	100	–	–	100	–	–	100	ns
TSC Input Delay	t_{PCT}		10	–	–	10	–	–	10	–	–	ns



(NOTE) Waveform measurements for all inputs and outputs are specified at logic "High" = V_{IHmin} and logic "Low" = V_{ILmax} unless otherwise specified

Figure 1 Read Data from Memory or Peripherals



(NOTE) Waveform measurements for all inputs and outputs are specified at logic "High" = V_{IHmin} and logic "Low" = V_{ILmax} unless otherwise specified

Figure 2 Write Data to Memory or Peripherals

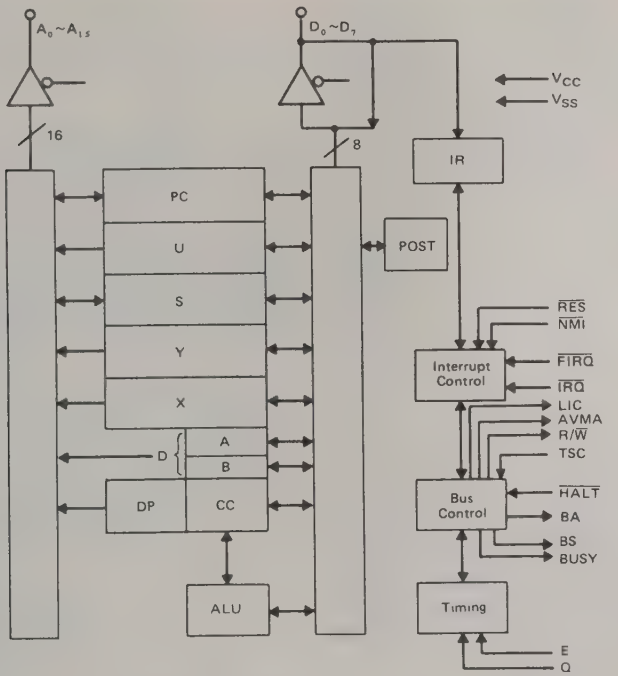
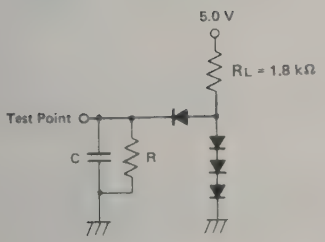


Figure 3 HD6309E Expanded Block Diagram



C = 30 pF for BA, BS, LIC, AVMA, BUSY
130 pF for $D_0 \sim D_7$
90 pF for $A_0 \sim A_{15}$, R/$\overline{W}$

R = 10 kΩ for $D_0 \sim D_7$
10 kΩ for $A_0 \sim A_{15}$, R/$\overline{W}$
10 kΩ for BA, BS, LIC, AVMA, BUSY

All diodes are 1S2074(H) or equivalent.
C includes stray capacitance.

Figure 4 Bus Timing Test Load

■ PROGRAMMING MODEL

As shown in Figure 5, the HD6309E adds three registers to the set available in the HD6800. The added registers include a Direct Page Register, the User Stack pointer and a second Index Register.

● Accumulators (A, B, D)

The A and B registers are general purpose accumulators which are used for arithmetic calculations and manipulation of data.

Certain instructions concatenate the A and B registers to form a single 16-bit accumulator. This is referred to as the D Register, and is formed with the A Register as the most significant byte.

● Direct Page Register (DP)

The Direct Page Register of the HD6309E serves to enhance the Direct Addressing Mode. The content of this register appears at the higher address outputs ($A_8 \sim A_{15}$) during direct addressing instruction execution. This allows the direct mode to be used at any place in memory, under program control. To ensure HD6800 compatibility, all bits of this register are cleared during Processor Reset.

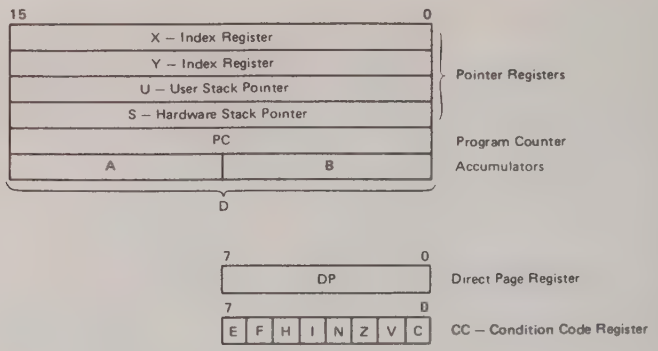


Figure 5 Programming Model of The Microprocessing Unit

- **Index Registers (X, Y)**

The Index Registers are used in indexed mode of addressing. The 16-bit address in this register takes part in the calculation of effective addresses. This address may be used to point to data directly or may be modified by an optional constant or register offset. During some indexed modes, the contents of the index register are incremented or decremented to point to the next item of tabular type data. All four pointer registers (X, Y, U, S) may be used as index registers.

- **Stack Pointer (U, S)**

The Hardware Stack Pointer (S) is used automatically by the processor during subroutine calls and interrupts. The User Stack Pointer (U) is controlled exclusively by the programmer thus allowing arguments to be passed to and from subroutines with ease. The U-register is frequently used as a stack marker. Both Stack Pointers have the same indexed mode addressing capabilities as the X and Y registers, but also support **Push** and **Pull** instructions. This allows the HD6309E to be used efficiently as a stack processor, greatly enhancing its ability to support higher level languages and modular programming.

(NOTE) The stack pointers of the HD6309E point to the top of the stack, in contrast to the HD6800 stack pointer, which pointed to the next free location on stack.

- **Program Counter (PC)**

The Program Counter is used by the processor to point to the address of the next instruction to be executed by the processor. Relative Addressing is provided allowing the Program Counter to be used like an index register in some situations.

- **Condition Code Register (CC)**

The Condition Code Register defines the state of the processor at any given time. See Figure 6.

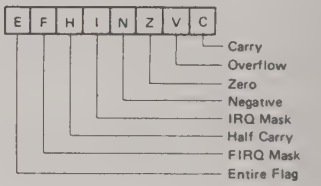


Figure 6 Condition Code Register Format

■ CONDITION CODE REGISTER DESCRIPTION

- **Bit 0 (C)**

Bit 0 is the carry flag, and is usually the carry from the binary ALU. C is also used to represent a 'borrow' from subtract like instructions (CMP, NEG, SUB, SBC) and is the complement of the carry from the binary ALU.

- **Bit 1 (V)**

Bit 1 is the overflow flag, and is set to a one by an operation which causes a signed two's complement arithmetic overflow. This overflow is detected in an operation in which the carry from the MSB in the ALU does not match the carry from the MSB-1.

- **Bit 2 (Z)**

Bit 2 is the zero flag, and is set to a one if the result of the previous operation was identically zero.

- **Bit 3 (N)**

Bit 3 is the negative flag, which contains exactly the value of the MSB of the result of the preceding operation. Thus, a negative two's-complement result will leave N set to a one.

- **Bit 4 (I)**

Bit 4 is the $\overline{IRQ}$ mask bit. The processor will not recognize interrupts from the $\overline{IRQ}$ line if this bit is set to a one. $\overline{NMI}$, $\overline{FIRQ}$, $\overline{IRQ}$, $\overline{RES}$ and SWI all set I to a one; SWI2 and SWI3 do not affect I.

- **Bit 5 (H)**

Bit 5 is the half-carry bit, and is used to indicate a carry from bit 3 in the ALU as a result of an 8-bit addition only (ADC or ADD). This bit is used by the DAA instruction to perform a BCD decimal add adjust operation. The state of this flag is undefined in all subtract-like instructions.

- **Bit 6 (F)**

Bit 6 is the $\overline{FIRQ}$ mask bit. The processor will not recognize interrupts from the $\overline{FIRQ}$ line if this bit is a one. $\overline{NMI}$, $\overline{FIRQ}$, SWI, and $\overline{RES}$ all set F to a one. $\overline{IRQ}$, SWI2 and SWI3 do not affect F.

- **Bit 7 (E)**

Bit 7 is the entire flag, and when set to a one indicates that the complete machine state (all the registers) was stacked, as opposed to the subset state (PC and CC). The E bit of the stacked CC is used on a return from interrupt (RTI) to determine the extent of the unstacking. Therefore, the current E left in the Condition Code Register represents past action.

■ HD6309E MPU SIGNAL DESCRIPTION

- **Power (V_{SS}, V_{CC})**

Two pins are used to supply power to the part: V_{SS} is ground or 0 volts, while V_{CC} is +5.0 V ±10%.

- **Address Bus ($A_0 \sim A_{15}$)**

Sixteen pins are used to output address information from the MPU onto the Address Bus. When the processor does not require the bus for a data transfer, it will output address $FFFF_{16}$, R/$\overline{W}$ = "High", and BS = "Low"; this is a "dummy access" or $\overline{VMA}$ cycle. All address bus drivers are made high-impedance when output Bus Available (BA) is "High" or when TSC is asserted. Each pin will drive one Schottky TTL load or four LS TTL loads, and 90 pF. Refer to Figures 1 and 2.

- **Data Bus ($D_0 \sim D_7$)**

These eight pins provide communication with the system bi-directional data bus. Each pin will drive one Schottky TTL load or four LS TTL loads, and 130 pF.

- **Read/Write (R/$\overline{W}$)**

This signal indicates the direction of data transfer on the data bus. A "Low" indicates that the MPU is writing data onto the data bus. R/$\overline{W}$ is made high impedance when BA is "High" or when TSC is asserted. Refer to Figures 1 and 2.

- **$\overline{RES}$**

A "Low" level on this Schmitt-trigger input for greater than one bus cycle will reset the MPU, as shown in Figure 7. The Reset vectors are fetched from locations $FFFE_{16}$ and $FFFF_{16}$ (Table 1) when Interrupt Acknowledge is true, ($\overline{BA}$ · BS = 1). During initial power-on, the Reset line should be held "Low" until the clock input signals are fully operational.

Because the HD6309E Reset pin has a Schmitt-trigger input with a threshold voltage higher than that of standard peripherals, a simple R/C network may be used to reset the entire system. This higher threshold voltage ensures that all peripherals are out of the reset state before the Processor.

Table 1 Memory Map for Interrupt Vectors

Memory Map for Vector Locations		Interrupt Vector Description
MS	LS	
FFFE	FFFF	$\overline{RES}$
FFFC	FFFD	$\overline{NMI}$
FFFA	FFFB	SWI
FFF8	FFF9	$\overline{IRQ}$
FFF6	FFF7	$\overline{FIRQ}$
FFF4	FFF5	SWI2
FFF2	FFF3	SWI3
FFF0	FFF1	Reserved

- **$\overline{HALT}$**

A "Low" level on this input pin will cause the MPU to stop running at the end of the present instruction and remain halted indefinitely without loss of data. When halted, the BA output is driven "High" indicating the buses are high impedance. BS is also "High" which indicates the processor is in the Halt state. While halted, the MPU will not respond to external real-time requests ($\overline{FIRQ}$, $\overline{IRQ}$) although $\overline{NMI}$ or $\overline{RES}$ will be latched for later response. During the Halt state Q and E should continue to run normally. A halted state (BA · BS = 1) can be achieved by pulling $\overline{HALT}$ "Low" while $\overline{RES}$ is still "Low". See Figure 8.

- **Bus Available, Bus Status (BA, BS)**

The Bus Available output is an indication of an internal control signal which makes the MOS buses of the MPU high impedance. When BA goes "Low", a dead cycle will elapse before the MPU acquires the bus. BA will not be asserted when TSC is active, thus allowing dead cycle consistency.

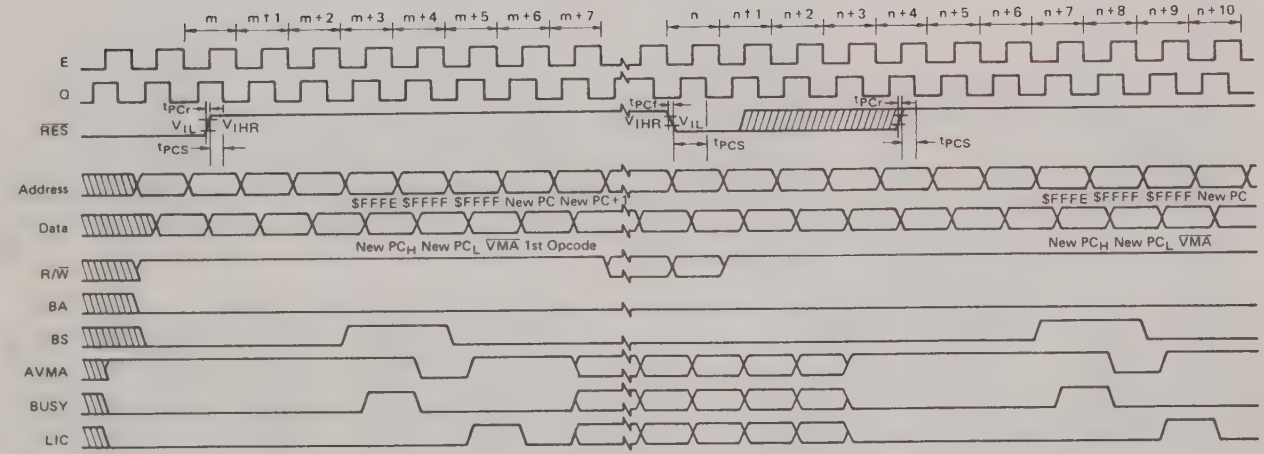
The Bus Status output signal, when decoded with BA, represents the MPU state (valid with leading edge of Q).

MPU State		MPU State Definition
BA	BS	
0	0	Normal (Running)
0	1	Interrupt or RESET Acknowledge
1	0	SYNC Acknowledge
1	1	HALT Acknowledge

Interrupt Acknowledge is indicated during both cycles of a hardware-vector-fetch ($\overline{RES}$, $\overline{NMI}$, $\overline{FIRQ}$, $\overline{IRQ}$, SWI, SWI2, SWI3). This signal, plus decoding of the lower four address lines, can provide the user with an indication of which interrupt level is being serviced and allow vectoring by device. See Table 1.

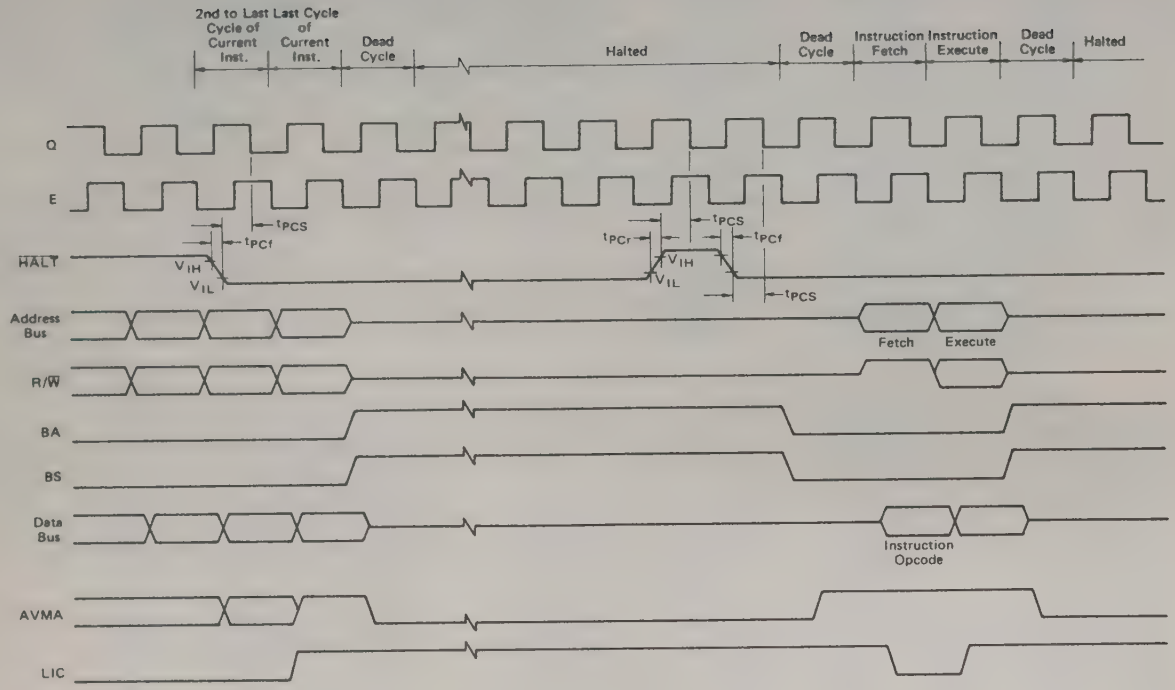
Sync Acknowledge is indicated while the MPU is waiting for external synchronization on an interrupt line.

Halt Acknowledge is indicated when the HD6309E is in a Halt condition.



(NOTE) Waveform measurements for all inputs and outputs are specified at logic "High" = V_{IHmin} and logic "Low" = V_{ILmax} unless otherwise specified.

Figure 7 $\overline{RES}$ Timing



(NOTE) Waveform measurements for all inputs and outputs are specified at logic "High" = V_{IHmin} and logic "Low" = V_{ILmax} unless otherwise specified.

Figure 8 $\overline{\text{HALT}}$ and Single Instruction Execution for System Debug

- **Non Maskable Interrupt ($\overline{NMI}$)***

A negative transition on this input requests that a non-maskable interrupt sequence be generated. A non-maskable interrupt cannot be inhibited by the program, and also has a higher priority than $\overline{FIRQ}$, $\overline{IRQ}$ or software interrupts. During recognition of an $\overline{NMI}$, the entire machine state is saved on the hardware stack. After reset, an $\overline{NMI}$ will not be recognized until the first program load of the Hardware Stack Pointer (S). The pulse width of $\overline{NMI}$ low must be at least one E cycle. If the $\overline{NMI}$ input does not meet the minimum set up with respect to Q, the interrupt will not be recognized until the next cycle. See Figure 9.

- **Fast-Interrupt Request ($\overline{FIRQ}$)***

A "Low" level on this input pin will initiate a fast interrupt sequence, provided its mask bit (F) in the CC is clear. This sequence has priority over the standard Interrupt Request ($\overline{IRQ}$), and is fast in the sense that it stacks only the contents of the condition code register and the program counter. The interrupt service routine should clear the source of the interrupt before doing an RTI. See Figure 10.

- **Interrupt Request ($\overline{IRQ}$)***

A "Low" level input on this pin will initiate an Interrupt Request sequence provided the mask bit (I) in the CC is clear. Since $\overline{IRQ}$ stacks the entire machine state it provides a slower response to interrupts than $\overline{FIRQ}$. $\overline{IRQ}$ also has a lower priority than $\overline{FIRQ}$. Again, the interrupt service routine should clear the source of the interrupt before doing an RTI. See Figure 9.

* $\overline{NMI}$, $\overline{FIRQ}$, and $\overline{IRQ}$ requests are sampled on the falling edge of Q. One cycle is required for synchronization before these interrupts are recognized. The pending interrupt(s) will not be serviced until completion of the current instruction unless a SYNC or CWAI condition is present. If $\overline{IRQ}$ and $\overline{FIRQ}$ do not remain "Low" until completion of the current instruction they may not be recognized. However, $\overline{NMI}$ is latched and need only remain "Low" for one cycle.

- **Clock Inputs E, Q**

E and Q are the clock signals required by the HD6309E. Q must lead E; that is, a transition on Q must be followed by a similar transition on E after a minimum delay. Addresses will be valid from the MPU, t_{AD} after the falling edge of E, and data will be latched from the bus by the falling edge of E. While the Q input is fully TTL compatible, the E input directly drives internal MOS circuitry and, thus, requires levels above normal TTL levels. This approach minimizes clock skew inherent with an internal buffer. Timing and waveforms for E and Q are shown in Figures 1 and 2 while Figure 11 shows a simple clock generator for the HD6309E.

- **BUSY**

Busy will be "High" for the read and modify cycles of a read-modify-write instruction and during the access of the first byte of a double-byte operation (e.g., LDX, STD, ADDD). Busy is also "High" during the first byte of any indirect or other vector fetch (e.g., jump extended, SWI indirect etc.).

In a multi-processor system, busy indicates the need to defer the rearbitration of the next bus cycle to insure the integrity of the above operations. This difference provides the indivisible memory access required for a "test-and-set" primitive, using any one of several read-modify-write instructions.

Busy does not become active during PSH or PUL operations. A typical read-modify-write instruction (ASL) is shown in Figure 12. Timing information is given in Figure 13. Busy is valid t_{CD} after the rising edge of Q.

- **AVMA**

AVMA is the Advanced VMA signal and indicates that the MPU will use the bus in the following bus cycle. The predictive nature of the AVMA signal allows efficient shared-bus multi-processor systems. AVMA is "Low" when the MPU is in either a HALT or SYNC state. AVMA is valid t_{CD} after the rising edge of Q.

- **LIC**

LIC (Last Instruction Cycle) is "High" during the last cycle of every instruction, and its transition from "High" to "Low" will indicate that the first byte of an opcode will be latched at the end of the present bus cycle. LIC will be "High" when the MPU is Halted at the end of an instruction, (i.e., not in CWAI or RESET) in SYNC state or while stacking during interrupts. LIC is valid t_{CD} after the rising edge of Q.

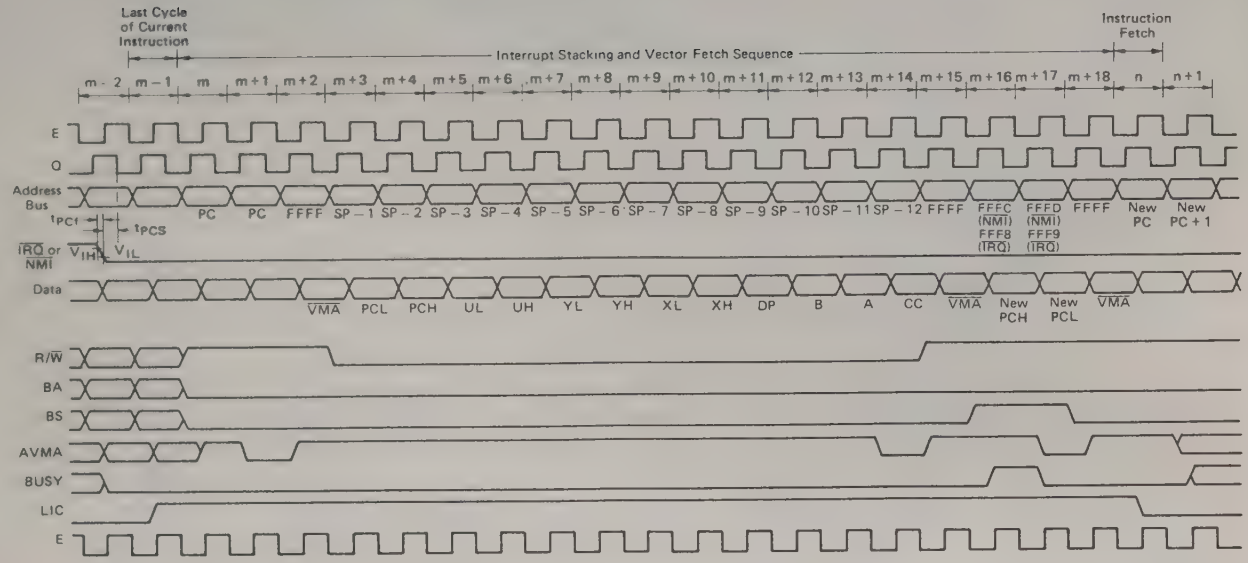
- **TSC**

TSC (Three-State Control) will cause MOS address, data, and R/$\overline{W}$ buffers to assume a high-impedance state. The control signals (BA, BS, BUSY, AVMA and LIC) will not go to the high-impedance state. TSC is intended to allow a single bus to be shared with other bus masters (processors or DMA controllers).

While E is "Low", TSC controls the address buffers and R/$\overline{W}$ directly. The data bus buffers during a write operation are in a high-impedance state until Q rises at which time, if TSC is true, they will remain in a high-impedance state. If TSC is held beyond the rising edge of E, then it will be internally latched, keeping the bus drivers in a high-impedance state for the remainder of the bus cycle. See Figure 14.

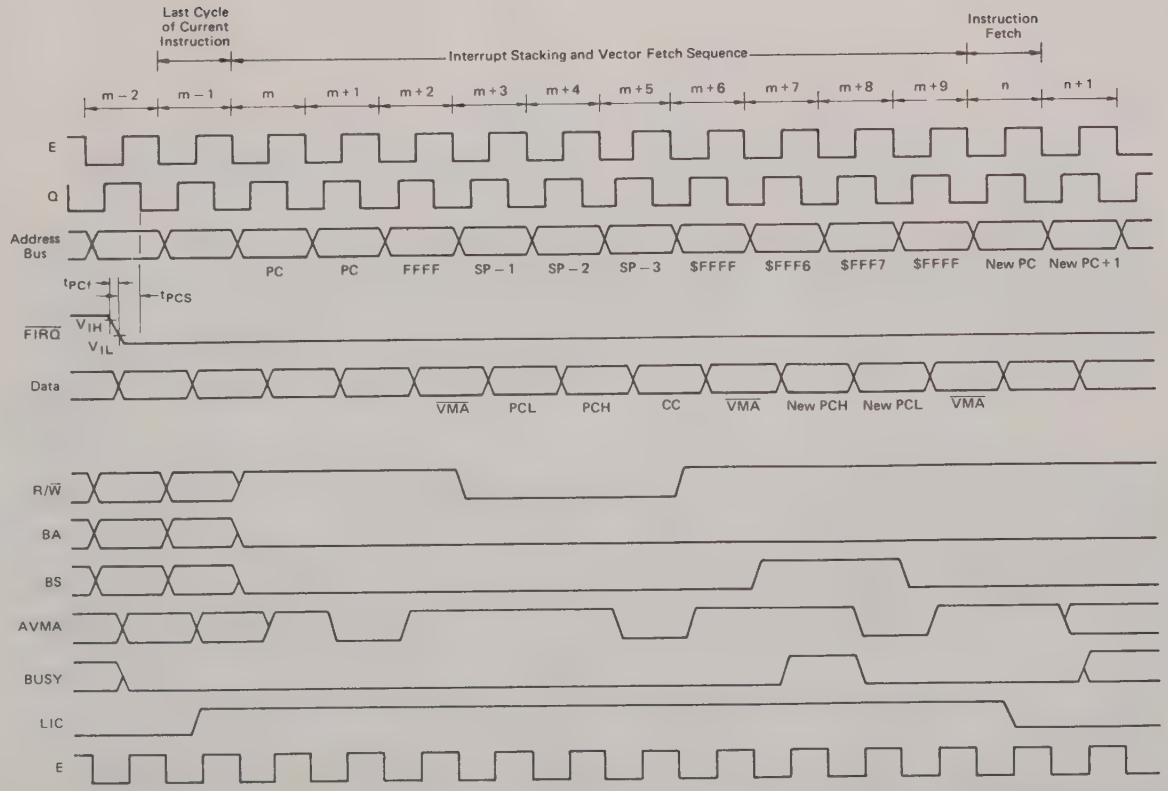
- **MPU Operation**

During normal operation, the MPU fetches an instruction from memory and then executes the requested function. This sequence begins after $\overline{RES}$ and is repeated indefinitely unless altered by a special instruction or hardware occurrence. Software instructions that alter normal MPU operation are: SWI, SWI2, SWI3, CWAI, RTI and SYNC. An interrupt or $\overline{HALT}$ input can also alter the normal execution of instructions. Figure 15 illustrates the flow chart for the HD6309E.


Last Cycle of Current Instruction
Interrupt Stacking and Vector Fetch Sequence
Instruction Fetch
m-2 m-1 m m+1 m+2 m+3 m+4 m+5 m+6 m+7 m+8 m+9 m+10 m+11 m+12 m+13 m+14 m+15 m+16 m+17 m+18 n n+1
E
Q
Address Bus
PC PC FFFF SP-1 SP-2 SP-3 SP-4 SP-5 SP-6 SP-7 SP-8 SP-9 SP-10 SP-11 SP-12 FFFF FFFC (NMI) FFF8 (IRQ) FFFD (NMI) FFF9 (IRQ) FFFF New PC New PC+1
tPCf
tPCS
IRQ or NMI
VIH
VIL
Data
VMA PCL PCH UL UH YL YH XL XH DP B A CC VMA New PCH New PCL VMA
R/W
BA
BS
AVMA
BUSY
LIC
E

(NOTE) Waveform measurements for all inputs and outputs are specified at logic "High" = V_{IHmin} and logic "Low" = V_{ILmax} unless otherwise specified.
E clock shown for reference only.

Figure 9 $\overline{IRQ}$ and $\overline{NMI}$ Interrupt Timing



(NOTE) Waveform measurements for all inputs and outputs are specified at logic "High" = V_{IHmin} and logic "Low" = V_{ILmax} unless otherwise specified. E clock shown for reference only.

Figure 10 $\overline{\text{FIRQ}}$ Interrupt Timing

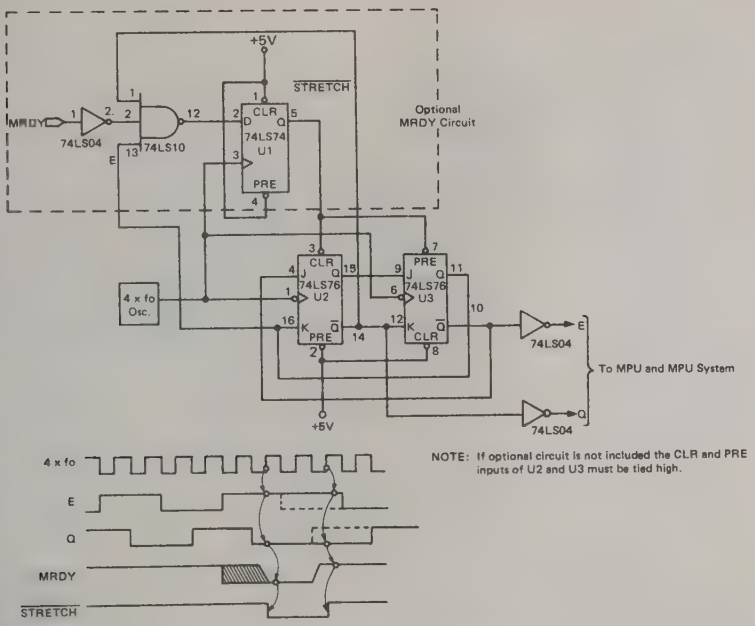


Figure 11 HD6309E Clock Generator

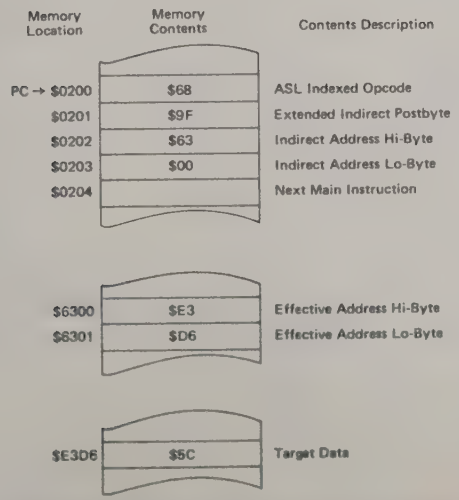
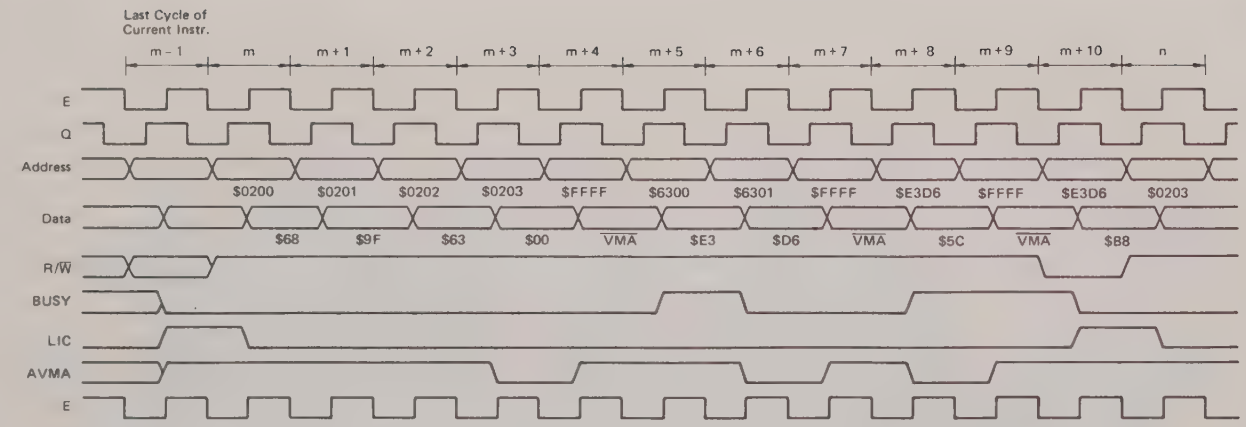
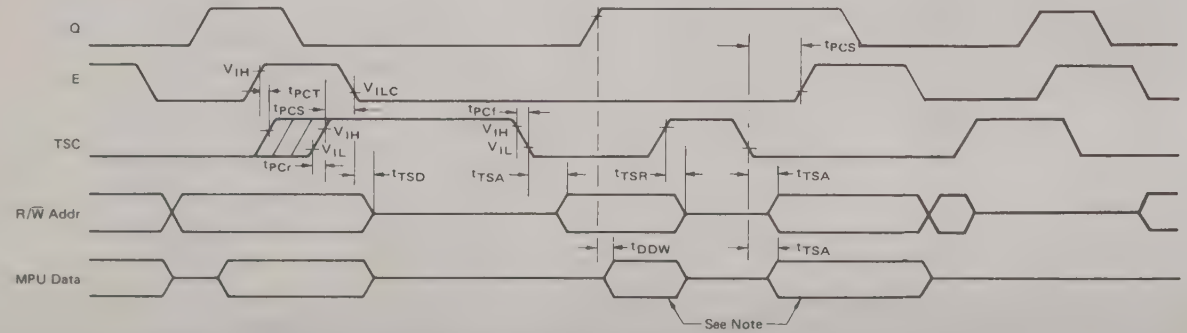


Figure 12 Read Modify Write Instruction Example (ASL Extended Indirect)



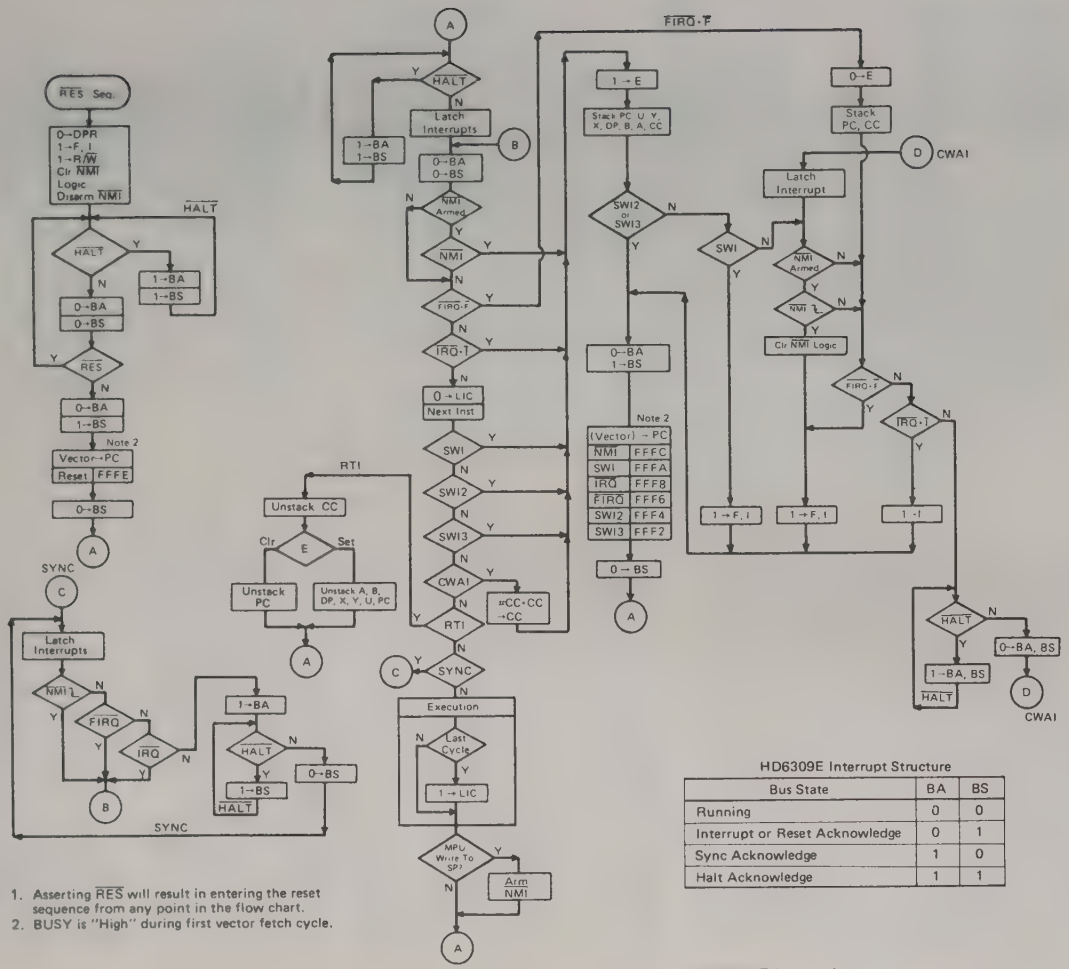
(NOTE) Waveform measurements for all inputs and outputs are specified at logic "High" = V_{IHmin} and logic "Low" = V_{ILmax} unless otherwise specified.

Figure 13 BUSY Timing (ASL Extended Indirect Instruction)



(NOTES) Data will be asserted by the MPU only during the interval while R/W is "Low" and E or Q is "High".
Waveform measurements for all inputs and outputs are specified at logic "High" = V_{IHmin} and logic "Low" = V_{ILmax} unless otherwise specified.

Figure 14 TSC Timing



HD6309E Interrupt Structure

Bus State	BA	BS
Running	0	0
Interrupt or Reset Acknowledge	0	1
Sync Acknowledge	1	0
Halt Acknowledge	1	1

(NOTES) 1. Asserting $\overline{RES}$ will result in entering the reset sequence from any point in the flow chart.
2. BUSY is "High" during first vector fetch cycle.

Figure 15 Flowchart for HD6309E Instruction

■ ADDRESSING MODES

The basic instructions of any computer are greatly enhanced by the presence of powerful addressing modes. The HD6309E has the most complete set of addressing modes available on any microcomputer today. For example, the HD6309E has 59 basic instructions; however, it recognizes 1464 different variations of instructions and addressing modes. The addressing modes support modern programming techniques. The following addressing modes are available on the HD6309E:

(1) Implied (Includes Accumulator)
(2) Immediate
(3) Extended
(4) Extended Indirect
(5) Direct
(6) Register
(7) Indexed
    - Zero-Offset
    - Constant Offset
    - Accumulator Offset
    - Auto Increment/Decrement
(8) Indexed Indirect
(9) Relative
(10) Program Counter Relative

• Implied (Includes Accumulator)

In this addressing mode, the opcode of the instruction contains all the address information necessary. Examples of Implied Addressing are: ABX, DAA, SWI, ASRA, and CLRB.

• Immediate Addressing

In Immediate Addressing, the effective address of the data is the location immediately following the opcode (i.e., the data to be used in the instruction immediately follows the opcode of the instruction). The HD6309E uses both 8 and 16-bit immediate values depending on the size of argument specified by the opcode. Examples of instructions with Immediate Addressing are:

```
LDA  #$20
LDX  #$F000
LDY  #CAT
```

(NOTE) # signifies immediate addressing, $ signifies hexadecimal value.

• Extended Addressing

In Extended Addressing, the contents of the two bytes immediately following the opcode fully specify the 16-bit effective address used by the instruction. Note that the address generated by an extended instruction defines an absolute address and is not position independent. Examples of Extended Addressing include:

```
LDA  CAT
STX  MOUSE
LDD  $2000
```

• Extended Indirect

As a special case of indexed addressing (discussed below), one level of indirection may be added to Extended Addressing. In Extended Indirect, the two bytes following the postbyte of an Indexed instruction contain the address of the data.

```
LDA  [CAT]
LDX  [$FFFE]
STU  [DOG]
```

• Direct Addressing

Direct addressing is similar to extended addressing except that only one byte of address follows the opcode. This byte specifies the lower 8 bits of the address to be used. The upper 8 bits of the addres, are supplied by the direct page register. Since only one byte of address is required in direct addressing, this mode requires less memory and executes faster than extended addressing. Of course, only 256 locations (one page) can be accessed without redefining the contents of the DP register. Since the DP register is set to $00 on Reset, direct addressing on the HD6309E is compatible with direct addressing on the HD6800. Indirection is not allowed in direct addressing. Some examples of direct addressing are:

```
LDA   $30
SETDP $10     (Assembler directive)
LDB   $1030
LDD   <CAT
```

(NOTE) < is an assembler directive which forces direct addressing.

• Register Addressing

Some opcodes are followed by a byte that defines a register or set of registers to be used by the instruction. This is called a postbyte. Some examples of register addressing are:

```
TFR   X, Y        Transfer X into Y
EXG   A, B        Exchanges A with B
PSHS  A, B, X, Y  Push Y, X, B and A onto S
PULU  X, Y, D     Pull D, X, and Y from U
```

• Indexed Addressing

In all indexed addressing, one of the pointer registers (X, Y, U, S, and sometimes PC) is used in a calculation of the effective address of the operand to be used by the instruction. Five basic types of indexing are available and are discussed below. The postbyte of an indexed instruction specifies the basic type and variation of the addressing mode as well as the pointer register to be used. Figure 16 lists the legal formats for the postbyte. Table 2 gives the assembler form and the number of cycles and bytes added to the basic values for indexed addressing for each variation.

Post-Byte Register Bit								Indexed Addressing Mode
7	6	5	4	3	2	1	0	
0	R	R	d	d	d	d	d	EA = ,R + 5 Bit Offset
1	R	R	0	0	0	0	0	,R +
1	R	R	1	0	0	0	1	,R + +
1	R	R	0	0	0	1	0	, -R
1	R	R	1	0	0	1	1	, - - R
1	R	R	1	0	1	0	0	EA = ,R + 0 Offset
1	R	R	1	0	1	0	1	EA = ,R + ACCB Offset
1	R	R	1	0	1	1	0	EA = ,R + ACCA Offset
1	R	R	1	1	0	0	0	EA = ,R + 8 Bit Offset
1	R	R	1	1	0	0	1	EA = ,R + 16 Bit Offset
1	R	R	1	1	0	1	1	EA = ,R + D Offset
1	x	x	1	1	1	0	0	EA = ,PC + 8 Bit Offset
1	x	x	1	1	1	0	1	EA = ,PC + 16 Bit Offset
1	R	R	1	1	1	1	1	EA = [,Address]

Addressing Mode Field
Indirect Field (Sign bit when b7 = 0)
Register Field RR: 00 = X, 01 = Y, 10 = U, 11 = S

x = Don't Care
d = Offset Bit
1 = { 0 = Non Indirect, 1 = Indirect }

Figure 16 Index Addressing Postbyte Register Bit Assignments

Table 2 Indexed Addressing Mode

Type	Forms	Non Indirect				Indirect			
		Assembler Form	Postbyte OP Code	+ ~	+ #	Assembler Form	Postbyte OP Code	+ ~	+ #
Constant Offset From R (2's Complement Offsets)	No Offset	,R	1RR00100	0	0	[,R]	1RR10100	3	0
	5 Bit Offset	n, R	0RRnnnnn	1	0	defaults to 8-bit			
	8 Bit Offset	n, R	1RR01000	1	1	[n, R]	1RR11000	4	1
	16 Bit Offset	n, R	1RR01001	4	2	[n, R]	1RR11001	7	2
Accumulator Offset From R (2's Complement Offsets)	A Register Offset	A, R	1RR00110	1	0	[A, R]	1RR10110	4	0
	B Register Offset	B, R	1RR00101	1	0	[B, R]	1RR10101	4	0
	D Register Offset	D, R	1RR01011	4	0	[D, R]	1RR11011	7	0
Auto Increment/Decrement R	Increment By 1	,R +	1RR00000	2	0	not allowed			
	Increment By 2	,R + +	1RR00001	3	0	[,R + +]	1RR10001	6	0
	Decrement By 1	, - R	1RR00010	2	0	not allowed			
	Decrement By 2	, - - R	1RR00011	3	0	[, - - R]	1RR10011	6	0
Constant Offset From PC (2's Complement Offsets)	8 Bit Offset	n, PCR	1xx01100	1	1	[n, PCR]	1xx11100	4	1
	16 Bit Offset	n, PCR	1xx01101	5	2	[n, PCR]	1xx11101	8	2
Extended Indirect	16 Bit Address	–	–	–	–	[n]	10011111	5	2

R = X, Y, U or S
x = Don't Care

RR:
00 = X
01 = Y
10 = U
11 = S

$\overset{+}{\sim}$ and $\overset{+}{\#}$ indicate the number of additional cycles and bytes for the particular variation.

Zero-Offset Indexed

In this mode, the selected pointer register contains the effective address of the data to be used by the instruction. This is the fastest indexing mode.

Examples are:

```
LDD   0, X
LDA   S
```

Constant Offset Indexed

In this mode, a two's-complement offset and the contents of one of the pointer registers are added to form the effective address of the operand. The pointer register's initial content is unchanged by the addition.

Three sizes of offsets are available:

5-bit (−16 to +15)
8-bit (−128 to +127)
16-bit (−32768 to +32767)

The two's complement 5-bit offset is included in the postbyte and, therefore, is most efficient in use of bytes and cycles. The two's complement 8-bit offset is contained in a single byte following the postbyte. The two's complement 16-bit offset is in the two bytes following the postbyte. In most cases the programmer need not be concerned with the size of this offset since the assembler will select the optimal size automatically.

Examples of constant-offset indexing are:

```
LDA   23, X
LDX   −2, S
LDY   300, X
LDU   CAT, Y
```

Accumulator-Offset Indexed

This mode is similar to constant offset indexed except that the two's-complement value in one of the accumulators (A, B or D) and the contents of one of the pointer registers are added to form the effective address of the operand. The contents of both the accumulator and the pointer register are unchanged by the addition. The postbyte specifies which accumulator to use as an offset and no additional bytes are required. The advantage of an accumulator offset is that the value of the offset can be calculated by a program at run-time.

Some examples are:

```
LDA   B, Y
LDX   D, Y
LEAX  B, X
```

Auto Increment/Decrement Indexed

In the auto increment addressing mode, the pointer register contains the address of the operand. Then, after the pointer register is used it is incremented by one or two. This addressing mode is useful in stepping through tables, moving data, or for the creation of software stacks. In auto decrement, the pointer register is decremented prior to use as the address of the data. The use of auto decrement is similar to that of auto increment; but the tables, etc., are scanned from the high to low addresses. The size of the increment/decrement can be either one or two to allow for tables of either 8- or 16-bit data to be accessed and is selectable by the programmer. The pre-

decrement, post-increment nature of these modes allow them to be used to create additional software stacks that behave identically to the U and S stacks.

Some examples of the auto increment/decrement addressing modes are:

```
LDA   ,X+
STD   ,Y++
LDB   ,-Y
LDX   ,--S
```

Care should be taken in performing operations on 16-bit pointer registers (X, Y, U, S) where the same register is used to calculate the effective address.

Consider the following instruction:

STX 0, X++ (X initialized to 0)

The desired result is to store a 0 in locations $0000 and $0001 then increment X to point to $0002. In reality, the following occurs:

0 → temp	calculate the EA; temp is a holding register
X + 2 → X	perform autoincrement
X → (temp)	do store operation

- **Indexed Indirect**

All of the indexing modes with the exception of auto increment/decrement by one, or a ±4-bit offset may have an additional level of indirection specified. In indirect addressing, the effective address is contained at the location specified by the contents of the Index Register plus any offset. In the example below, the A accumulator is loaded indirectly using an effective address calculated from the Index Register and an offset.

```
                Before Execution
                A = XX (don't care)
                X = $F000
$0100   LDA [$10, X]      EA is now $F010
$F010   $F1               $F150 is now the
$F011   $50               new EA
$F150   $AA
                After Execution
                A = $AA (Actual Data Loaded)
                X = $F000
```

All modes of indexed indirect are included except those which are meaningless (e.g., auto increment/decrement by 1 indirect). Some examples of indexed indirect are:

```
LDA   [,X]
LDD   [10,S]
LDA   [B,Y]
LDD   [,X++]
```

- **Relative Addressing**

The byte(s) following the branch opcode is (are) treated as a signed offset which may be added to the program counter. If the branch condition is true then the calculated address (PC + signed offset) is loaded into the program counter. Program execution continues at the new location as indicated by the PC; short (1 byte offset) and long (2 bytes offset) relative addressing modes are available. All of memory can be reached in long relative addressing as an effective address is interpreted modulo 2^{16}. Some examples of relative addressing are:

```
        BEQ     CAT     (short)
        BGT     DOG     (short)

CAT     LBEQ    RAT     (long)
DOG     LBGT    RABBIT  (long)
        .
        .
        .
RAT     NOP
RABBIT  NOP
```

- **Program Counter Relative**

The PC can be used as the pointer register with 8 or 16-bit signed offsets. As in relative addressing, the offset is added to the current PC to create the effective address. The effective address is then used as the address of the operand or data. Program Counter Relative Addressing is used for writing position independent programs. Tables related to a particular routine will maintain the same relationship after the routine is moved, if referenced relative to the Program Counter. Examples are:

```
LDA   CAT,PCR
LEAX  TABLE,PCR
```

Since program counter relative is a type of indexing, an additional level of indirection is available.

```
LDA   [CAT,PCR]
LDU   [DOG,PCR]
```

■ HD6309E INSTRUCTION SET

The instruction set of the HD6309E is similar to that of the HD6800 and is upward compatible at the source code level. The number of opcodes has been reduced from 72 to 59, but because of the expanded architecture and additional addressing modes, the number of available opcodes (with different addressing modes) has risen from 197 to 1464.

Some of the instructions are described in detail below:

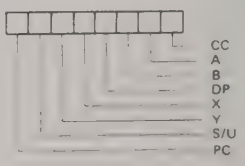
- **PSHU/PSHS**

The push instructions have the capability of pushing onto either the hardware stack (S) or user stack (U) any single register, or set of registers with a single instruction.

- **PULU/PULS**

The pull instructions have the same capability of the push instruction, in reverse order. The byte immediately following the push or pull opcode determines which register or registers are to be pushed or pulled. The actual PUSH/PULL sequence is fixed; each bit defines a unique register to push or pull, as shown in below.

PUSH/PULL POST BYTE




```
                ← Pull Order          Push Order →
PC              U   Y   X   DP  B   A   CC
FFFF ..... ← increasing memory address ..... 0000
PC              S   Y   X   DP  B   A   CC
```

- **TFR/EXG**

Within the HD6309E, any register may be transferred to or exchanged with another of like-size; i.e., 8-bit to 8-bit or 16-bit to 16-bit. Bits 4~7 of postbyte define the source register, while bits 0~3 represent the destination register. These are denoted as follows:

0000 – D	0101 – PC
0001 – X	1000 – A
0010 – Y	1001 – B
0011 – U	1010 – CC
0100 – S	1011 – DP

(NOTE) All other combinations are undefined and INVALID.

TRANSFER/EXCHANGE POST BYTE

- **LEAX/LEAY/LEAU/LEAS**

The LEA (Load Effective Address) works by calculating the effective address used in an indexed instruction and stores that address value, rather than the data at that address, in a pointer register. This makes all the features of the internal addressing hardware available to the programmer. Some of the implications of this instruction are illustrated in Table 3.

The LEA instruction also allows the user to access data in a position independent manner. For example:

```
        LEAX    MSG1, PCR
        LBSR    PDATA (Print message routine)
        .
        .
MSG1    FCC     'MESSAGE'
```

This sample program prints: 'MESSAGE'. By writing MSG1, PCR, the assembler computes the distance between the present address and MSG1. This result is placed as a constant into the LEAX instruction which will be indexed from the PC value at the time of execution. No matter where the code is located, when it is executed, the computed offset from the PC will put the absolute address of MSG1 into the X pointer register. This code is totally position independent.

The LEA instructions are very powerful and use an internal holding register (temp). Care must be exercised when using the LEA instructions with the autoincrement and autodecrement addressing modes due to the sequence of internal operations. The LEA internal sequence is outlined as follows:

```
LEAa, b+        (any of the 16-bit pointer registers X, Y, U
                 or S may be substituted for a and b.)
1. b → temp     (calculate the EA)
2. b + 1 → b    (modify b, postincrement)
3. temp → a     (load a)

LEAa, – b
1. b – 1 → temp (calculate EA with predecrement)
2. b – 1 → b    (modify b, predecrement)
3. temp → a     (load a)
```

Autoincrement-by-two and autodecrement-by-two instructions work similarly. Note that LEAX, X+ does not change X, however LEAX, –X does decrement X. LEAX 1, X should be used to increment X by one.

Table 3 LEA Examples

Instruction	Operation	Comment
LEAX 10, X	X + 10 → X	Adds 5-bit constant 10 to X
LEAX 500, X	X + 500 → X	Adds 16-bit constant 500 to X
LEAY A, Y	Y + A → Y	Adds 8-bit A accumulator to Y
LEAY D, Y	Y + D → Y	Adds 16-bit D accumulator to Y
LEAU –10, U	U – 10 → U	Subtracts 10 from U
LEAS –10, S	S – 10 → S	Used to reserve area on stack
LEAS 10, S	S + 10 → S	Used to 'clean up' stack
LEAX 5, S	S + 5 → X	Transfers as well as adds

- **MUL**

Multiplies the unsigned binary numbers in the A and B accumulator and places the unsigned result into the 16-bit D accumulator. This unsigned multiply also allows multiple-precision multiplications.

Long and Short Relative Branches

The HD6309E has the capability of program counter relative branching throughout the entire memory map. In this mode, if the branch is to be taken, the 8 or 16-bit signed offset is added to the value of the program counter to be used as the effective address. This allows the program to branch anywhere in the 64k memory map. Position independent code can be easily generated through the use of relative branching. Both short (8-bit) and long (16-bit) branches are available.

- **SYNC**

After encountering a Sync instruction, the MPU enters a Sync state, stops processing instructions and waits for an interrupt. If the pending interrupt is non-maskable ($\overline{NMI}$) or maskable ($\overline{FIRQ}$, $\overline{IRQ}$) with its mask bit (F or I) clear, the processor will clear the Sync state and perform the normal interrupt stacking and service routine. Since $\overline{FIRQ}$ and $\overline{IRQ}$ are not edge-triggered, a low level with a minimum duration of three bus cycles is required to assure that the interrupt will be taken. If the pending interrupt is maskable ($\overline{FIRQ}$, $\overline{IRQ}$) with its mask bit (F or I) set, the processor will clear the Sync state and continue processing by executing the next inline instruction. Figure 17 depicts Sync timing.

Software Interrupts

A Software Interrupt is an instruction which will cause an interrupt, and its associated vector fetch. These Software Interrupts are useful in operating system calls, software debugging, trace operations, memory mapping, and software development systems. Three levels of SWI are available on this HD6309E, and are prioritized in the following order: SWI, SWI2, SWI3.

16-Bit Operation

The HD6309E has the capability of processing 16-bit data. These instructions include loads, stores, compares, adds, subtracts, transfers, exchanges, pushes and pulls.

- **CYCLE-BY-CYCLE OPERATION**

The address bus cycle-by-cycle performance chart illustrates the memory-access sequence corresponding to each possible instruction and addressing mode in the HD6309E. Each instruction begins with an opcode fetch. While that opcode is being internally decoded, the next program byte is always fetched. (Most instructions will use the next byte, so this

technique considerably speeds throughput.) Next, the operation of each opcode will follow the flow chart. $\overline{VMA}$ is an indication of $FFFF_{16}$ on the address bus, R/$\overline{W}$ = "High" and BS = "Low". The following examples illustrate the use of the chart; see Figure 18.

Example 1: LBSR (Branch Taken)

Before Execution SP = F000

```
                .
                .
                .
$8000           LBSR    CAT
                .
                .
                .
$A000   CAT     .
```

CYCLE-BY-CYCLE FLOW

Cycle #	Address	Data	R/$\overline{W}$	Description
1	8000	17	1	Opcode Fetch
2	8001	1F	1	Offset High Byte
3	8002	FD	1	Offset Low Byte
4	FFFF	*	1	$\overline{VMA}$ Cycle
5	FFFF	*	1	$\overline{VMA}$ Cycle
6	FFFF	*	1	$\overline{VMA}$ Cycle
7	FFFF	*	1	$\overline{VMA}$ Cycle
8	EFFF	03	0	Stack Low Order Byte of Return Address
9	EFFE	80	0	Stack High Order Byte of Return Address

Example 2: DEC (Extended)

```
$8000   DEC     $A000
$A000   FCB     $80
```

CYCLE-BY-CYCLE FLOW

Cycle #	Address	Data	R/$\overline{W}$	Description
1	8000	7A	1	Opcode Fetch
2	8001	A0	1	Operand Address, High Byte
3	8002	00	1	Operand Address, Low Byte
4	FFFF	*	1	$\overline{VMA}$ Cycle
5	A000	80	1	Read the Data
6	FFFF	*	1	$\overline{VMA}$ Cycle
7	A000	7F	0	Store the Decremented Data

* The data bus has the data at that particular address.

■ SLEEP MODE

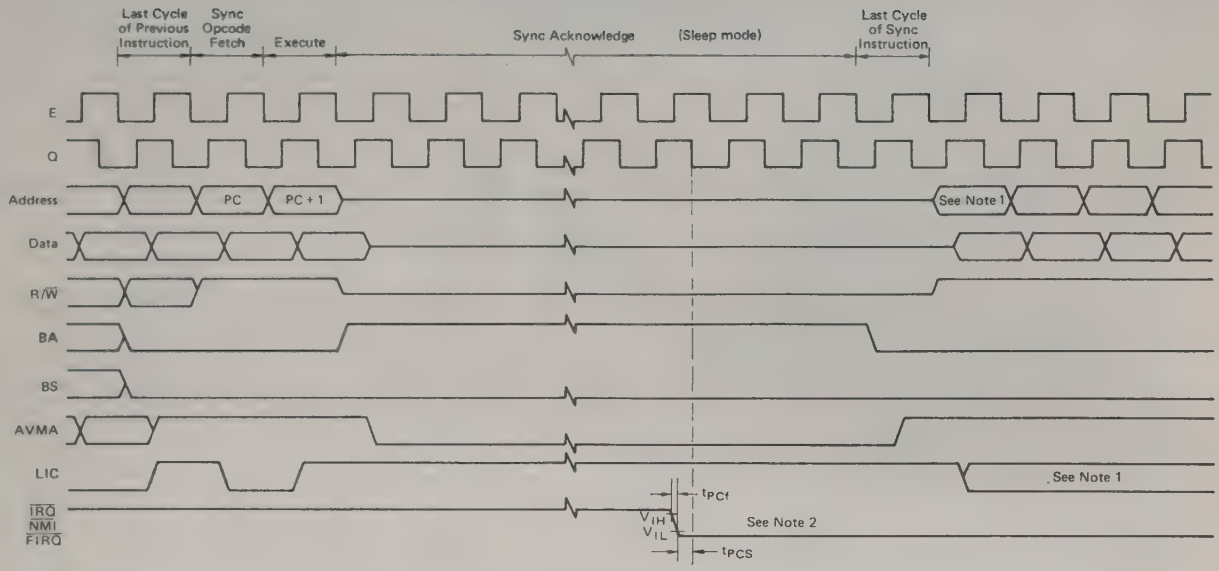
During the interrupt wait period in the SYNC instruction (the SYNC state) and that period in the CWAI instruction (the WAIT state), MPU operation is halted and goes to the sleep mode. However, the state of I/O pins is the same as that of the HD6809E in this mode.

■ HD6309E INSTRUCTION SET TABLES

The instructions of the HD6309E have been broken down into five different categories. They are as follows:

8-Bit operation (Table 4)
16-Bit operation (Table 5)
Index register/stack pointer instructions (Table 6)
Relative branches (long or short) (Table 7)
Miscellaneous instructions (Table 8)

HD6309E instruction set tables and Hexadecimal Values of instructions are shown in Table 9 and Table 10.

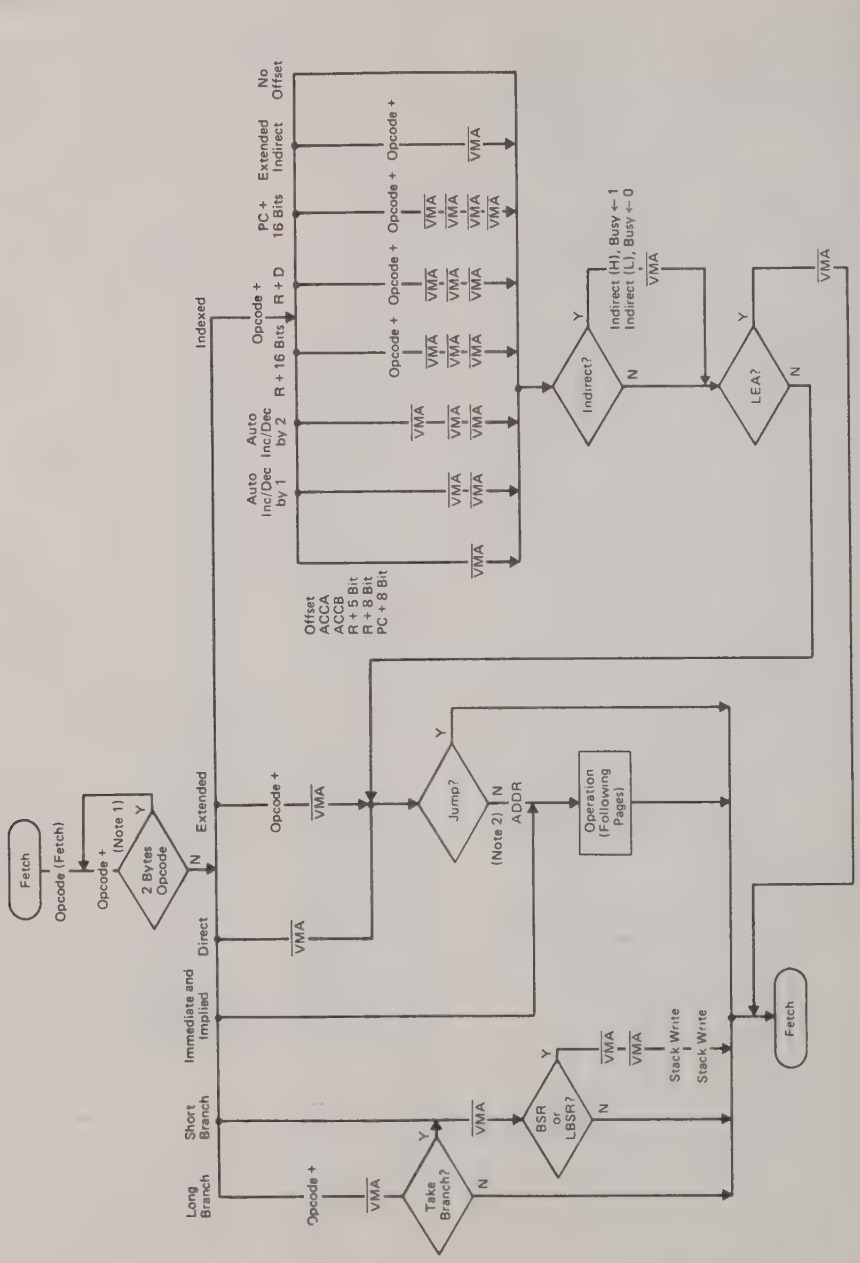


(NOTES) 1. If the associated mask bit is set when the interrupt is requested, LIC will go "Low" and this cycle will be an instruction fetch from address location PC + 1. However, if the interrupt is accepted ($\overline{NMI}$ or an unmasked $\overline{FIRQ}$ or $\overline{IRQ}$) LIC will remain "High" and interrupt processing will start with this cycle as (m) on Figure 9 and 10 (Interrupt Timing).

2. If mask bits are clear, $\overline{IRQ}$ and $\overline{FIRQ}$ must be held "Low" for three cycles to guarantee that interrupt will be taken, although only one cycle is necessary to bring the processor out of SYNC.

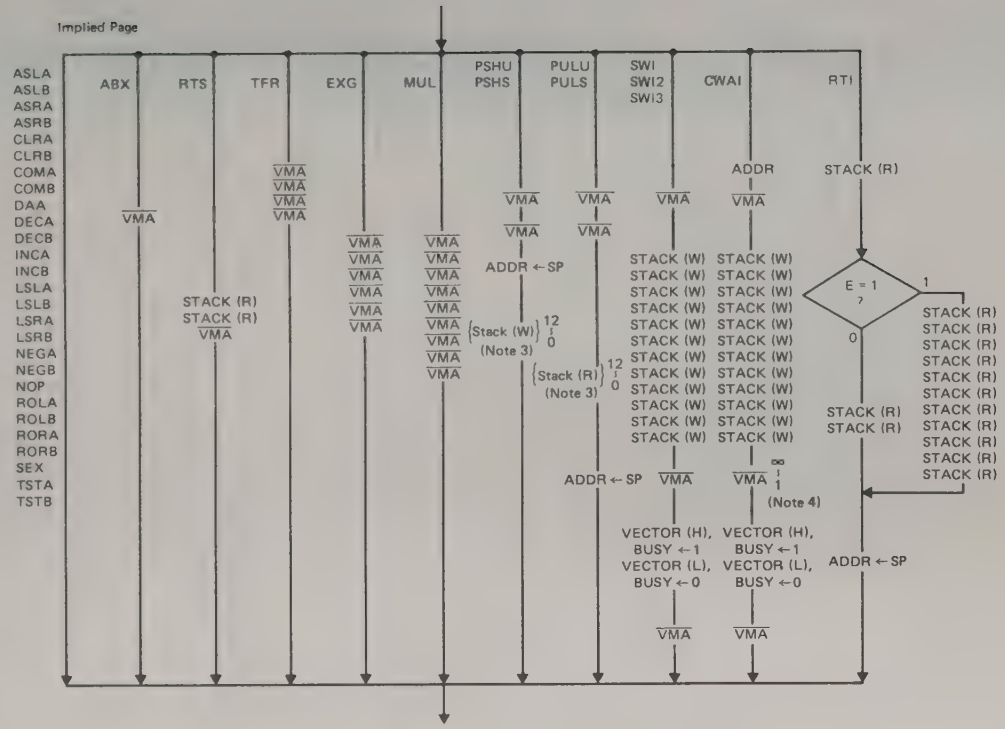
3. Waveform measurements for all inputs and outputs are specified at logic "High" = V_{IHmin} and logic "Low" = V_{ILmax} unless otherwise specified.

Figure 17 SYNC Timing



(NOTE)
1. Busy = "High" during access of first byte of double byte immediate load.
2. Write operation during store instruction. Busy = "High" during first two cycles of a double-byte access and the first cycle of read-modify-write access.
3. AVMA is asserted on the cycle before a VMA cycle.

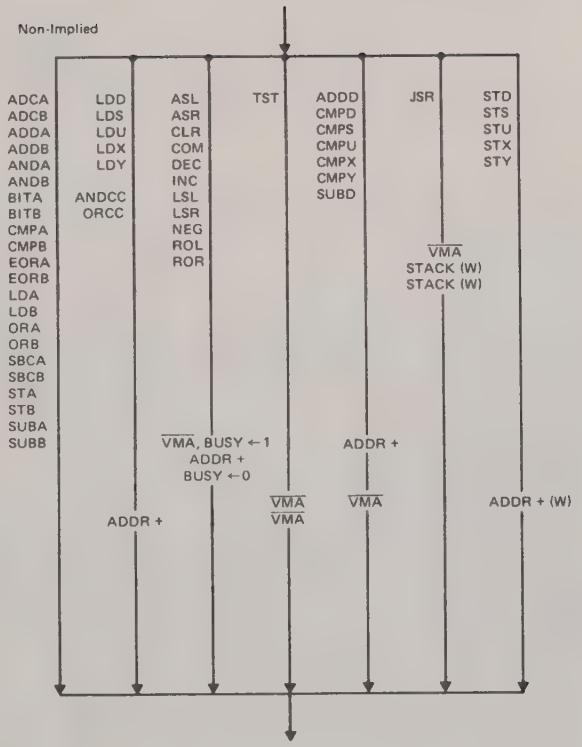
Figure 18 Address Bus Cycle-by-Cycle Performance



(NOTES)

1. Stack (W) refers to the following sequence: SP ← SP – 1, then ADDR ← SP with R/$\overline{W}$ = "Low"
   Stack (R) refers to the following sequence: ADDR ← SP with R/$\overline{W}$ = "High", then SP ← SP + 1.
   PSHU, PULU instructions use the user stack pointer (i.e., SP = U) and PSHS, PULS use the hardware stack pointer (i.e., SP = S).
2. Vector refers to the address of an interrupt or reset vector (see Table 1).
3. The number of stack accesses will vary according to the number of bytes saved.
4. $\overline{VMA}$ cycles will occur until an interrupt occurs.

Figure 18 Address Bus Cycle-by-Cycle Performance (Continued)



(NOTES)

1. Stack (W) refers to the following sequence: SP ← SP − 1, then ADDR ← SP with R/$\overline{W}$ = "Low"
   Stack (R) refers to the following sequence: ADDR ← SP with R/$\overline{W}$ = "High", then SP ← SP + 1.
   PSHU, PULU instructions use the user stack pointer (i.e., SP = U) and PSHS, PULS use the hardware stack pointer (i.e., SP = S).
2. Vector refers to the address of an interrupt or reset vector (see Table 1).
3. The number of stack accesses will vary according to the number of bytes saved.
4. $\overline{VMA}$ cycles will occur until an interrupt occurs.

Figure 18 Address Bus Cycle-by-Cycle Performance (Continued)

Table 4 8-Bit Accumulator and Memory Instructions

Mnemonic(s)	Operation
ADCA, ADCB	Add memory to accumulator with carry
ADDA, ADDB	Add memory to accumulator
ANDA, ANDB	And memory with accumulator
ASL, ASLA, ASLB	Arithmetic shift of accumulator or memory left
ASR, ASRA, ASRB	Arithmetic shift of accumulator or memory right
BITA, BITB	Bit test memory with accumulator
CLR, CLRA, CLRB	Clear accumulator or memory location
CMPA, CMPB	Compare memory from accumulator
COM, COMA, COMB	Complement accumultor or memory location
DAA	Decimal adjust A accumulator
DEC, DECA, DECB	Decrement accumulator or memory location
EORA, EORB	Exclusive or memory with accumulator
EXG R1, R2	Exchange R1 with R2 (R1, R2 = A, B, CC, DP)
INC, INCA, INCB	Increment accumulator or memory location
LDA, LDB	Load accumulator from memory
LSL, LSLA, LSLB	Logical shift left accumulator or memory location
LSR, LSRA, LSRB	Logical shift right accumulator or memory location
MUL	Unsigned multiply (A × B → D)
NEG, NEGA, NEGB	Negate accumulator or memory
ORA, ORB	Or memory with accumulator
ROL, ROLA, ROLB	Rotate accumulator or memory left
ROR, RORA, RORB	Rotate accumulator or memory right
SBCA, SBCB	Subtract memory from accumulator with borrow
STA, STB	Store accumulator to memory
SUBA, SUBB	Subtract memory from accumulator
TST, TSTA, TSTB	Test accumulator or memory location
TFR R1, R2	Transfer R1 to R2 (R1, R2 = A, B, CC, DP)

(NOTE) A, B, CC or DP may be pushed to (pulled from) either stack with PSHS, PSHU (PULS, PULU) instructions.

Table 5 16-Bit Accumulator and Memory Instructions

Mnemonic(s)	Operation
ADDD	Add memory to D accumulator
CMPD	Compare memory from D accumulator
EXG D, R	Exchange D with X, Y, S, U or PC
LDD	Load D accumulator from memory
SEX	Sign Extend B accumulator into A accumulator
STD	Store D accumulator to memory
SUBD	Subtract memory from D accumulator
TFR D, R	Transfer D to X, Y, S, U or PC
TFR R, D	Transfer X, Y, S, U or PC to D

(NOTE) D may be pushed (pulled) to either stack with PSHS, PSHU (PULS, PULU) instructions.

Table 6 Index Register Stack Pointer Instructions

Mnemonic(s)	Operation
CMPS, CMPU	Compare memory from stack pointer
CMPX, CMPY	Compare memory from index register
EXG R1, R2	Exchange D, X, Y, S, U or PC with D, X, Y, S, U or PC
LEAS, LEAU	Load effective address into stack pointer
LEAX, LEAY	Load effective address into index register
LDS, LDU	Load stack pointer from memory
LDX, LDY	Load index register from memory
PSHS	Push A, B, CC, DP, D, X, Y, U, or PC onto hardware stack
PSHU	Push A, B, CC, DP, D, X, Y, S, or PC onto user stack
PULS	Pull A, B, CC, DP, D, X, Y, U or PC from hardware stack
PULU	Pull A, B, CC, DP, D, X, Y, S or PC from user stack
STS, STU	Store stack pointer to memory
STX, STY	Store index register to memory
TFR R1, R2	Transfer D, X, Y, S, U or PC to D, X, Y, S, U or PC
ABX	Add B accumulator to X (unsigned)

Table 7 Branch Instructions

Mnemonic(s)	Operation
	SIMPLE BRANCHES
BEQ, LBEQ	Branch if equal
BNE, LBNE	Branch if not equal
BMI, LBMI	Branch if minus
BPL, LBPL	Branch if plus
BCS, LBCS	Branch if carry set
BCC, LBCC	Branch if carry clear
BVS, LBVS	Branch if overflow set
BVC, LBVC	Branch if overflow clear
	SIGNED BRANCHES
BGT, LBGT	Branch if greater (signed)
BGE, LBGE	Branch if greater than or equal (signed)
BEQ, LBEQ	Branch if equal
BLE, LBLE	Branch if less than or equal (signed)
BLT, LBLT	Branch if less than (signed)
	UNSIGNED BRANCHES
BHI, LBHI	Branch if higher (unsigned)
BHS, LBHS	Branch if higher or same (unsigned)
BEQ, LBEQ	Branch if equal
BLS, LBLS	Branch if lower or same (unsigned)
BLO, LBLO	Branch if lower (unsigned)
	OTHER BRANCHES
BSR, LBSR	Branch to subroutine
BRA, LBRA	Branch always
BRN, LBRN	Branch never

Table 8 Miscellaneous Instructions

Mnemonic(s)	Operation
ANDCC	AND condition code register
CWAI	AND condition code register, then wait for interrupt
NOP	No operation
ORCC	OR condition code register
JMP	Jump
JSR	Jump to subroutine
RTI	Return from interrupt
RTS	Return from subroutine
SWI, SWI2, SWI3	Software interrupt (absolute indirect)
SYNC	Synchronize with interrupt line

Table 9 HD6309E Instruction Set Table

INSTRUCTION/ FORMS		HD6309E ADDRESSING MODES																	DESCRIPTION	5	3	2	1	0
		IMPLIED			DIRECT			EXTENDED			IMMEDIATE			INDEXED①			RELATIVE				H	N	Z	V	C
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~⑤	#						
ABX		3A	3	1																B + X → X (UNSIGNED)	•	•	•	•	•
ADC	ADCA				99	4	2	B9	5	3	89	2	2	A9	4+	2+				A + M + C → A	↕	↕	↕	↕	↕
	ADCB				D9	4	2	F9	5	3	C9	2	2	E9	4+	2+				B + M + C → B	↕	↕	↕	↕	↕
ADD	ADDA				9B	4	2	BB	5	3	8B	2	2	AB	4+	2+				A + M → A	↕	↕	↕	↕	↕
	ADDB				DB	4	2	FB	5	3	CB	2	2	EB	4+	2+				B + M → B	↕	↕	↕	↕	↕
	ADDD				D3	6	2	F3	7	3	C3	4	3	E3	6+	2+				D + M:M + 1 → D	•	↕	↕	↕	↕
AND	ANDA				94	4	2	B4	5	3	84	2	2	A4	4+	2+				A ∧ M → A	•	↕	↕	0	•
	ANDB				D4	4	2	F4	5	3	C4	2	2	E4	4+	2+				B ∧ M → B	•	↕	↕	0	•
	ANDCC										1C	3	2							CC ∧ IMM → CC	(	—	7	—	)
ASL	ASLA	48	2	1																A	8	↕	↕	↕	↕
	ASLB	58	2	1																B	8	↕	↕	↕	↕
	ASL				08	6	2	78	7	3				68	6+	2+				M	8	↕	↕	↕	↕
ASR	ASRA	47	2	1																A	8	↕	↕	•	↕
	ASRB	57	2	1																B	8	↕	↕	•	↕
	ASR				07	6	2	77	7	3				67	6+	2+				M	8	↕	↕	•	↕
BCC	BCC																24	3	2	Branch C = 0	•	•	•	•	•
	LBCC																10	5(6)	4	Long Branch	•	•	•	•	•
																	24			C = 0					
BCS	BCS																25	3	2	Branch C = 1	•	•	•	•	•
	LBCS																10	5(6)	4	Long Branch	•	•	•	•	•
																	25			C = 1					
BEQ	BEQ																27	3	2	Branch Z = 1	•	•	•	•	•
	LBEQ																10	5(6)	4	Long Branch	•	•	•	•	•
																	27			Z = 1					
BGE	BGE																2C	3	2	Branch N ⊕ V=0	•	•	•	•	•
	LBGE																10	5(6)	4	Long Branch	•	•	•	•	•
																	2C			N ⊕ V=0					
BGT	BGT																2E	3	2	Branch Z∨(N⊕V)=0	•	•	•	•	•
	LBGT																10	5(6)	4	Long Branch	•	•	•	•	•
																	2E			Z∨(N ⊕ V)=0					
BHI	BHI																22	3	2	Branch C∨Z=0	•	•	•	•	•
	LBHI																10	5(6)	4	Long Branch	•	•	•	•	•
																	22			C∨Z=0					
BHS	BHS																24	3	2	Branch C=0	•	•	•	•	•
	LBHS																10	5(6)	4	Long Branch	•	•	•	•	•
																	24			C=0					
BIT	BITA				95	4	2	B5	5	3	85	2	2	A5	4+	2+				Bit Test A (M ∧ A)	•	↕	↕	0	•
	BITB				D5	4	2	F5	5	3	C5	2	2	E5	4+	2+				Bit Test B (M ∧ B)	•	↕	↕	0	•
BLE	BLE																2F	3	2	Branch Z∨(N⊕V)=1	•	•	•	•	•
	LBLE																10	5(6)	4	Long Branch	•	•	•	•	•
																	2F			Z∨(N ⊕ V)=1					
BLO	BLO																25	3	2	Branch C=1	•	•	•	•	•
	LBLO																10	5(6)	4	Long Branch	•	•	•	•	•
																	25			C=1					
BLS	BLS																23	3	2	Branch C∨Z=1	•	•	•	•	•
	LBLS																10	5(6)	4	Long Branch	•	•	•	•	•
																	23			C∨Z=1					
BLT	BLT																2D	3	2	Branch N ⊕ V=1	•	•	•	•	•
	LBLT																10	5(6)	4	Long Branch	•	•	•	•	•
																	2D			N ⊕ V=1					
BMI	BMI																2B	3	2	Branch N=1	•	•	•	•	•
	LBMI																10	5(6)	4	Long Branch	•	•	•	•	•
																	2B			N=1					
BNE	BNE																26	3	2	Branch Z = 0	•	•	•	•	•
	LBNE																10	5(6)	4	Long Branch	•	•	•	•	•
																	26			Z = 0					
BPL	BPL																2A	3	2	Branch N = 0	•	•	•	•	•
	LBPL																10	5(6)	4	Long Branch	•	•	•	•	•
																	2A			N = 0					
BRA	BRA																20	3	2	Branch Always	•	•	•	•	•
	LBRA																16	5	3	Long Branch/ Always	•	•	•	•	•
BRN	BRN																21	3	2	Branch Never	•	•	•	•	•
	LBRN																10	5	4	Long Branch Never	•	•	•	•	•
																	21								

(to be continued)

INSTRUCTION/ FORMS		HD6309E ADDRESSING MODES																DESCRIPTION	5	3	2	1	0
		IMPLIED			DIRECT			EXTENDED			IMMEDIATE			INDEXED[0]			RELATIVE				H	N	Z	V	C
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~[3]	#						
BSR	BSR																8D	7	2	Branch to Subroutine	•	•	•	•	•
	LBSR																17	9	3	Long Branch to Subroutine	•	•	•	•	•
BVC	BVC																28[1]	3	2	Branch V = 0	•	•	•	•	•
	LBVC																10 28	5(6)	4	Long Branch V = 0	•	•	•	•	•
BVS	BVS																29	3	2	Branch V = 1	•	•	•	•	•
	LBVS																10 29	5(6)	4	Long Branch V = 1	•	•	•	•	•
CLR	CLRA	4F	2	1																0 → A	•	0	1	0	0
	CLRB	5F	2	1																0 → B	•	0	1	0	0
	CLR				0F	6	2	7F	7	3				6F	6+	2+				0 → M	•	0	1	0	0
CMP	CMPA				91	4	2	B1	5	3	81	2	2	A1	4+	2+				Compare M from A	↕	↕	↕	↕	↕
	CMPB				D1	4	2	F1	5	3	C1	2	2	E1	4+	2+				Compare M from B	↕	↕	↕	↕	↕
	CMPD				10 93	7	3	10 B3	8	4	10 83	5	4	10 A3	7+	3+				Compare M: M + 1 from D	•	↕	↕	↕	↕
	CMPS				11 9C	7	3	11 BC	8	4	11 8C	5	4	11 AC	7+	3+				Compare M: M + 1 from S	•	↕	↕	↕	↕
	CMPU				11 93	7	3	11 B3	8	4	11 83	5	4	11 A3	7+	3+				Compare M: M + 1 from U	•	↕	↕	↕	↕
	CMPX				9C	6	2	BC	7	3	8C	4	3	AC	6+	2+				Compare M: M + 1 from X	•	↕	↕	↕	↕
	CMPY				10 9C	7	3	10 BC	8	4	10 8C	5	4	10 AC	7+	3+				Compare M: M + 1 from Y	•	↕	↕	↕	↕
COM	COMA	43	2	1																$\overline{A}$ → A	•	↕	↕	0	1
	COMB	53	2	1																$\overline{B}$ → B	•	↕	↕	0	1
	COM				03	6	2	73	7	3				63	6+	2+				$\overline{M}$ → M	•	↕	↕	0	1
CWAI		3C	20	2																CC ∧ IMM → CC (except 1→E) Wait for Interrupt	↕	—	—	7	—
DAA		19	2	1																Decimal Adjust A	•	↕	↕	8	↕
DEC	DECA	4A	2	1																A − 1 → A	•	↕	↕	↕	•
	DECB	5A	2	1																B − 1 → B	•	↕	↕	↕	•
	DEC				0A	6	2	7A	7	3				6A	6+	2+				M − 1 → M	•	↕	↕	↕	•
EOR	EORA				98	4	2	B8	5	3	88	2	2	A8	4+	2+				A ⊕ M → A	•	↕	↕	0	•
	EORB				D8	4	2	F8	5	3	C8	2	2	E8	4+	2+				B ⊕ M → B	•	↕	↕	0	•
EXG	R1, R2	1E	7	2																R1 ↔ R2[2]	—	—	—	10	—
INC	INCA	4C	2	1																A + 1 → A	•	↕	↕	↕	•
	INCB	5C	2	1																B + 1 → B	•	↕	↕	↕	•
	INC				0C	6	2	7C	7	3				6C	6+	2+				M + 1 → M	•	↕	↕	↕	•
JMP					0E	3	2	7E	4	3				6E	3+	2+				EA[4] → PC	•	•	•	•	•
JSR					9D	7	2	BD	8	3				AD	7+	2+				Jump to Subroutine	•	•	•	•	•
LD	LDA				96	4	2	B6	5	3	86	2	2	A6	4+	2+				M → A	•	↕	↕	0	•
	LDB				D6	4	2	F6	5	3	C6	2	2	E6	4+	2+				M → B	•	↕	↕	0	•
	LDD				DC	5	2	FC	6	3	CC	3	3	EC	5+	2+				M: M + 1 → D	•	↕	↕	0	•
	LDS				10 DE	6	3	10 FE	7	4	10 CE	4	4	10 EE	6+	3+				M: M + 1 → S	•	↕	↕	0	•
	LDU				DE	5	2	FE	6	3	CE	3	3	EE	5+	2+				M: M + 1 → U	•	↕	↕	0	•
	LDX				9E	5	2	BE	6	3	8E	3	3	AE	5+	2+				M: M + 1 → X	•	↕	↕	0	•
	LDY				10 9E	6	3	10 BE	7	4	10 8E	4	4	10 AE	6+	3+				M: M + 1 → Y	•	↕	↕	0	•
LEA	LEAS													32	4+	2+				EA[3] → S	•	•	•	•	•
	LEAU													33	4+	2+				EA[3] → U	•	•	•	•	•
	LEAX													30	4+	2+				EA[3] → X	•	•	↕	•	•
	LEAY													31	4+	2+				EA[3] → Y	•	•	↕	•	•
LSL	LSLA	48	2	1																A, B, M)	•	↕	↕	↕	↕
	LSLB	58	2	1																	•	↕	↕	↕	↕
	LSL				08	6	2	78	7	3				68	6+	2+					•	↕	↕	↕	↕
LSR	LSRA	44	2	1																A, B, M)	•	0	↕	•	↕
	LSRB	54	2	1																	•	0	↕	•	↕
	LSR				04	6	2	74	7	3				64	6+	2+					•	0	↕	•	↕
MUL		3D	11	1																A × B → D (Unsigned)	•	•	↕	•	⑨
NEG	NEGA	40	2	1																$\overline{A}$ + 1 → A	⑧	↕	↕	↕	↕
	NEGB	50	2	1																$\overline{B}$ + 1 → B	⑧	↕	↕	↕	↕
	NEG				00	6	2	70	7	3				60	6+	2+				$\overline{M}$ + 1 → M	⑧	↕	↕	↕	↕
NOP		12	2	1																No Operation	•	•	•	•	•

(to be continued)

INSTRUCTION/ FORMS		HD6309E ADDRESSING MODES																DESCRIPTION	5	3	2	1	0
		IMPLIED			DIRECT			EXTENDED			IMMEDIATE			INDEXED①			RELATIVE				H	N	Z	V	C
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~③	#						
OR	ORA				9A	4	2	BA	5	3	8A	2	2	AA	4+	2+				A ∨ M → A	•	↕	↕	0	•
	ORB				DA	4	2	FA	5	3	CA	2	2	EA	4+	2+				B ∨ M → B	•	↕	↕	0	•
	ORCC										1A	3	2							CC ∨ IMM → CC	(			7	)
PSH	PSHS	34	5+④	2																Push Registers on S Stack	•	•	•	•	•
	PSHU	36	5+④	2																Push Registers on U Stack	•	•	•	•	•
PUL	PULS	35	5+④	2																Pull Registers from S Stack	(			10	)
	PULU	37	5+④	2																Pull Registers from U Stack	(			10	)
ROL	ROLA	49	2	1																A	•	↕	↕	↕	↕
	ROLB	59	2	1																B	•	↕	↕	↕	↕
	ROL				09	6	2	79	7	3				69	6+	2+				M	•	↕	↕	↕	↕
ROR	RORA	46	2	1																A	•	↕	↕	•	↕
	RORB	56	2	1																B	•	↕	↕	•	↕
	ROR				06	6	2	76	7	3				66	6+	2+				M	•	↕	↕	•	↕
RTI		38	6/15	1																Return From Interrupt	(			7	)
RTS		39	5	1																Return From Subroutine	•	•	•	•	•
SBC	SBCA				92	4	2	B2	5	3	82	2	2	A2	4+	2+				A − M − C → A	⇕	↕	↕	↕	↕
	SBCB				D2	4	2	F2	5	3	C2	2	2	E2	4+	2+				B − M − C → B	⇕	↕	↕	↕	↕
SEX		1D	2	1																Sign Extend B into A	•	↕	↕	•	•
ST	STA				97	4	2	B7	5	3				A7	4+	2+				A → M	•	↕	↕	0	•
	STB				D7	4	2	F7	5	3				E7	4+	2+				B → M	•	↕	↕	0	•
	STD				DD	5	2	FD	6	3				ED	5+	2+				D → M: M+1	•	↕	↕	0	•
	STS				10 DF	6	3	10 FF	7	4				10 EF	6+	3+				S → M: M + 1	•	↕	↕	0	•
	STU				DF	5	2	FF	6	3				EF	5+	2+				U → M: M + 1	•	↕	↕	0	•
	STX				9F	5	2	BF	6	3				AF	5+	2+				X → M: M + 1	•	↕	↕	0	•
	STY				10 9F	6	3	10 BF	7	4				10 AF	6+	3+				Y → M: M + 1	•	↕	↕	0	•
SUB	SUBA				90	4	2	B0	5	3	80	2	2	A0	4+	2+				A − M → A	⇕	↕	↕	↕	↕
	SUBB				D0	4	2	F0	5	3	C0	2	2	E0	4+	2+				B − M → B	⇕	↕	↕	↕	↕
	SUBD				93	6	2	B3	7	3	83	4	3	A3	6+	2+				D − M: M + 1 → D	•	↕	↕	↕	↕
SWI	SWI⑥	3F	19	1																Software Interrupt1	•	•	•	•	•
	SWI2⑥	10 3F	20	2																Software Interrupt2	•	•	•	•	•
	SWI3⑥	11 3F	20	2																Software Interrupt3	•	•	•	•	•
SYNC		13	>2	1																Synchronize to Interrupt	•	•	•	•	•
TFR	R1, R2	1F	6	2																R1 → R2 ②	(			10	)
TST	TSTA	4D	2	1																Test A	•	↕	↕	0	•
	TSTB	5D	2	1																Test B	•	↕	↕	0	•
	TST				0D	6	2	7D	7	3				6D	6+	2+				Test M	•	↕	↕	0	•

(NOTES)

(1) This column gives a base cycle and byte count. To obtain total count, and the values obtained from the INDEXED ADDRESSING MODES table.
(2) R1 and R2 may be any pair of 8 bit or any pair of 16 bit registers.
The 8 bit registers are: A, B, CC, DP
The 16 bit registers are: X, Y, U, S, D, PC
(3) EA is the effective address.
(4) The PSH and PUL instructions require 5 cycle plus 1 cycle for each byte pushed or pulled.
(5) 5(6) means: 5 cycles if branch not taken, 6 cycles if taken.
(6) SWI sets I and F bits. SWI2 and SWI3 do not affect I and F.
(7) Conditions Codes set as a direct result of the instruction.
(8) Value of half-carry flag is undefined.
(9) Special Case – Carry set if b7 is SET.
(10) Condition Codes set as a direct result of the instruction if CC is specified, and not affected otherwise.

LEGEND:

OP	Operation Code (Hexadecimal)	Z	Zero (byte)
~	Number of MPU Cycles	V	Overflow, 2's complement
#	Number of Program Bytes	C	Carry from bit 7
+	Arithmetic Plus	↕	Test and set if true, cleared otherwise
−	Arithmetic Minus	•	Not Affected
×	Multiply	CC	Condition Code Register
M̄	Complement of M	:	Concatenation
→	Transfer Into	∨	Logical or
H	Half-carry (from bit 3)	∧	Logical and
N	Negative (sign bit)	⊕	Logical Exclusive or

Table 10 Hexadecimal Values of Machine Codes

OP	Mnem	Mode	~	#	OP	Mnem	Mode	~	#	OP	Mnem	Mode	~	#
00	NEG	Direct	6	2	30	LEAX	Indexed	4+	2+	60	NEG	Indexed	6+	2+
01	*				31	LEAY		4+	2+	61	*			
02	*				32	LEAS		4+	2+	62	*			
03	COM		6	2	33	LEAU	Indexed	4+	2+	63	COM		6+	2+
04	LSR		6	2	34	PSHS	Implied	5+	2	64	LSR		6+	2+
05	*				35	PULS		5+	2	65	*			
06	ROR		6	2	36	PSHU		5+	2	66	ROR		6+	2+
07	ASR		6	2	37	PULU		5+	2	67	ASR		6+	2+
08	ASL, LSL		6	2	38	*				68	ASL, LSL		6+	2+
09	ROL		6	2	39	RTS		5	1	69	ROL		6+	2+
0A	DEC		6	2	3A	ABX		3	1	6A	DEC		6+	2+
0B	*				3B	RTI		6, 15	1	6B	*			
0C	INC		6	2	3C	CWAI		20	2	6C	INC		6+	2+
0D	TST		6	2	3D	MUL		11	1	6D	TST		6+	2+
0E	JMP		3	2	3E	*				6E	JMP		3+	2+
0F	CLR	Direct	6	2	3F	SWI	Implied	19	1	6F	CLR	Indexed	6+	2+
10	See	—	—	—	40	NEGA	Implied	2	1	70	NEG	Extended	7	3
11	Next Page	—	—	—	41	*				71	*			
12	NOP	Implied	2	1	42	*				72	*			
13	SYNC	Implied	2	1	43	COMA		2	1	73	COM		7	3
14	*				44	LSRA		2	1	74	LSR		7	3
15	*				45	*				75	*			
16	LBRA	Relative	5	3	46	RORA		2	1	76	ROR		7	3
17	LBSR	Relative	9	3	47	ASRA		2	1	77	ASR		7	3
18	*				48	ASLA, LSLA		2	1	78	ASL, LSL		7	3
19	DAA	Implied	2	1	49	ROLA		2	1	79	ROL		7	3
1A	ORCC	Immed	3	2	4A	DECA		2	1	7A	DEC		7	3
1B	*	—			4B	*				7B	*			
1C	ANDCC	Immed	3	2	4C	INCA		2	1	7C	INC		7	3
1D	SEX	Implied	2	1	4D	TSTA		2	1	7D	TST		7	3
1E	EXG	↕	8	2	4E	*				7E	JMP		4	3
1F	TFR	Implied	6	2	4F	CLRA	Implied	2	1	7F	CLR	Extended	7	3
20	BRA	Relative	3	2	50	NEGB	Implied	2	1	80	SUBA	Immed	2	2
21	BRN		3	2	51	*				81	CMPA		2	2
22	BHI		3	2	52	*				82	SBCA		2	2
23	BLS		3	2	53	COMB		2	1	83	SUBD		4	3
24	BHS, BCC		3	2	54	LSRB		2	1	84	ANDA		2	2
25	BLO, BCS		3	2	55	*				85	BITA		2	2
26	BNE		3	2	56	RORB		2	1	86	LDA		2	2
27	BEQ		3	2	57	ASRB		2	1	87	*			
28	BVC		3	2	58	ASLB, LSLB		2	1	88	EORA		2	2
29	BVS		3	2	59	ROLB		2	1	89	ADCA		2	2
2A	BPL		3	2	5A	DECB		2	1	8A	ORA		2	2
2B	BMI		3	2	5B	*				8B	ADDA		2	2
2C	BGE		3	2	5C	INCB		2	1	8C	CMPX	Immed	4	3
2D	BLT		3	2	5D	TSTB		2	1	8D	BSR	Relative	7	2
2E	BGT		3	2	5E	*				8E	LDX	Immed	3	3
2F	BLE	Relative	3	2	5F	CLRB	Implied	2	1	8F	*			

LEGEND:
~ Number of MPU cycles (less possible push pull or indexed-mode cycles)
# Number of program bytes
* Denotes unused opcode

(to be continued)

OP	Mnem	Mode	~	#
90	SUBA	Direct	4	2
91	CMPA		4	2
92	SBCA		4	2
93	SUBD		6	2
94	ANDA		4	2
95	BITA		4	2
96	LDA		4	2
97	STA		4	2
98	EORA		4	2
99	ADCA		4	2
9A	ORA		4	2
9B	ADDA		4	2
9C	CMPX		6	2
9D	JSR		7	2
9E	LDX		5	2
9F	STX	Direct	5	2
A0	SUBA	Indexed	4+	2+
A1	CMPA		4+	2+
A2	SBCA		4+	2+
A3	SUBD		6+	2+
A4	ANDA		4+	2+
A5	BITA		4+	2+
A6	LDA		4+	2+
A7	STA		4+	2+
A8	EORA		4+	2+
A9	ADCA		4+	2+
AA	ORA		4+	2+
AB	ADDA		4+	2+
AC	CMPX		6+	2+
AD	JSR		7+	2+
AE	LDX		5+	2+
AF	STX	Indexed	5+	2+
B0	SUBA	Extended	5	3
B1	CMPA		5	3
B2	SBCA		5	3
B3	SUBD		7	3
B4	ANDA		5	3
B5	BITA		5	3
B6	LDA		5	3
B7	STA		5	3
B8	EORA		5	3
B9	ADCA		5	3
BA	ORA		5	3
BB	ADDA		5	3
BC	CMPX		7	3
BD	JSR		8	3
BE	LDX		6	3
BF	STX	Extended	6	3
C0	SUBB	Immed	2	2
C1	CMPB		2	2
C2	SBCB		2	2
C3	ADDD		4	3
C4	ANDB		2	2
C5	BITB	Immed	2	2
C6	LDB	Immed	2	2
C7	*			
C8	EORB		2	2
C9	ADCB		2	2
CA	ORB		2	2
CB	ADDB		2	2
CC	LDD		3	3
CD	*			
CE	LDU	Immed	3	3
CF	*			
D0	SUBB	Direct	4	2
D1	CMPB		4	2
D2	SBCB		4	2
D3	ADDD		6	2
D4	ANDB		4	2
D5	BITB		4	2
D6	LDB		4	2
D7	STB		4	2
D8	EORB		4	2
D9	ADCB		4	2
DA	ORB		4	2
DB	ADDB		4	2
DC	LDD		5	2
DD	STD		5	2
DE	LDU		5	2
DF	STU	Direct	5	2
E0	SUBB	Indexed	4+	2+
E1	CMPB		4+	2+
E2	SBCB		4+	2+
E3	ADDD		6+	2+
E4	ANDB		4+	2+
E5	BITB		4+	2+
E6	LDB		4+	2+
E7	STB		4+	2+
E8	EORB		4+	2+
E9	ADCB		4+	2+
EA	ORB		4+	2+
EB	ADDB		4+	2+
EC	LDD		5+	2+
ED	STD		5+	2+
EE	LDU		5+	2+
EF	STU	Indexed	5+	2+
F0	SUBB	Extended	5	3
F1	CMPB		5	3
F2	SBCB		5	3
F3	ADDD		7	3
F4	ANDB		5	3
F5	BITB		5	3
F6	LDB		5	3
F7	STB		5	3
F8	EORB		5	3
F9	ADCB		5	3
FA	ORB		5	3
FB	ADDB	Extended	5	3
FC	LDD	Extended	6	3
FD	STD		6	3
FE	LDU		6	3
FF	STU	Extended	6	3

2 Bytes Opcode

OP	Mnem	Mode	~	#
1021	LBRN	Relative	5	4
1022	LBHI		5(6)	4
1023	LBLS		5(6)	4
1024	LBHS, LBCC		5(6)	4
1025	LBCS, LBLO		5(6)	4
1026	LBNE		5(6)	4
1027	LBEQ		5(6)	4
1028	LBVC		5(6)	4
1029	LBVS		5(6)	4
102A	LBPL		5(6)	4
102B	LBMI		5(6)	4
102C	LBGE		5(6)	4
102D	LBLT		5(6)	4
102E	LBGT		5(6)	4
102F	LBLE	Relative	5(6)	4
103F	SWI2	Implied	20	2
1083	CMPD	Immed	5	4
108C	CMPY		5	4
108E	LDY	Immed	4	4
1093	CMPD	Direct	7	3
109C	CMPY		7	3
109E	LDY		6	3
109F	STY	Direct	6	3
10A3	CMPD	Indexed	7+	3+
10AC	CMPY		7+	3+
10AE	LDY		6+	3+
10AF	STY	Indexed	6+	3+
10B3	CMPD	Extended	8	4
10BC	CMPY		8	4
10BE	LDY		7	4
10BF	STY	Extended	7	4
10CE	LDS	Immed	4	4
10DE	LDS	Direct	6	3
10DF	STS	Direct	6	3
10EE	LDS	Indexed	6+	3+
10EF	STS	Indexed	6+	3+
10FE	LDS	Extended	7	4
10FF	STS	Extended	7	4
113F	SWI3	Implied	20	2
1183	CMPU	Immed	5	4
118C	CMPS	Immed	5	4
1193	CMPU	Direct	7	3
119C	CMPS	Direct	7	3
11A3	CMPU	Indexed	7+	3+
11AC	CMPS	Indexed	7+	3+
11B3	CMPU	Extended	8	4
11BC	CMPS	Extended	8	4

(NOTE): All unused opcodes are both undefined and illegal

■ NOTE FOR USE

Execution Sequence of CLR Instruction

Cycle-by-cycle flow of CLR instruction (Direct, Extended, Indexed Addressing Mode) is shown below. In this sequence the content of the memory location specified by the operand is read before writing "00" into it. Note that status Flags, such as IRQ Flag, will be cleared by this extra data read operation when accessing the control/status register (sharing the same address between read and write) of peripheral devices.

Example: CLR (Extended)

$8000	CLR	$A000
$A000	FCB	$80

Cycle #	Address	Data	R/$\overline{W}$	Description
1	8000	7F	1	Opcode Fetch
2	8001	A0	1	Operand Address, High Byte
3	8002	00	1	Operand Address, Low Byte
4	FFFF	*	1	$\overline{VMA}$ Cycle
5	A000	80	1	Read the Data
6	FFFF	*	1	$\overline{VMA}$ Cycle
7	A000	00	0	Store Fixed "00" into Specified Location

* The data bus has the data at that particular address.

HD6821, HD68A21, HD68B21—
PIA (Peripheral Interface Adapter)

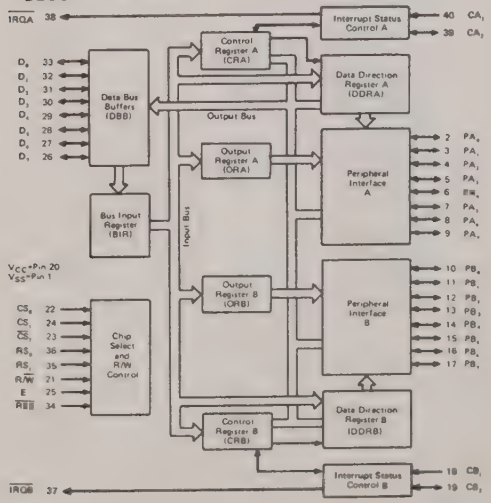
The HD6821 Peripheral Interface Adapter provides the universal means of interfacing peripheral equipment to the HD6800 Microprocessing Unit (MPU). This device is capable of interfacing the MPU to peripherals through two 8-bit bi-directional peripheral data buses and four control lines. No external logic is required for interfacing to most peripheral devices.

The functional configuration of the PIA is programmed by the MPU during system initialization. Each of the peripheral data lines can be programmed to act as an input or output, and each of the four control/interrupt lines may be programmed for one of several control modes. This allows a high degree of flexibility in the over-all operation of the interface.

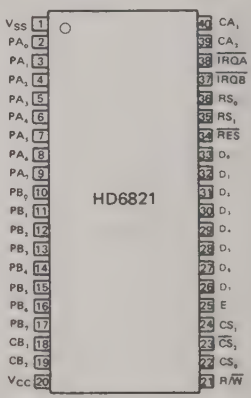
■ FEATURES

- Two Bi-directional 8-Bit Peripheral Data Bus for interface to Peripheral devices
- Two Programmable Control Registers
- Two Programmable Data Direction Registers
- Four Individually-Controlled Interrupt Input Lines: Two Usable as Peripheral Control Outputs
- Handshake Control Logic for Input and Output Peripheral Operation
- High-Impedance 3-State and Direct Transistor Drive Peripheral Lines
- Program Controlled Interrupt and Interrupt Disable Capability
- CMOS Drive Capability on Side A Peripheral Lines
- Two TTL Drive Capability on All A and B Side Buffers
- N Channel Silicon Gate MOS
- Compatible with MC6821, MC68A21 and MC68B21

■ BLOCK DIAGRAM



■ PIN ARRANGEMENT



(Top View)

■ ABSOLUTE MAXIMUM RATINGS

Item	Symbol	Value	Unit
Supply Voltage	V_{CC}*	-0.3 ~ +7.0	V
Input Voltage	V_{in}*	-0.3 ~ +7.0	V
Operating Temperature	T_{opr}	-20 ~ +75	°C
Storage Temperature	T_{stg}	-55 ~ +150	°C

* With respect to V_{SS} (SYSTEM GND)

(NOTE) Permanent LSI damage may occur if maximum ratings are exceeded. Normal operation should be under recommended operating conditions. If these conditions are exceeded, it could affect reliability of LSI.

■ RECOMMENDED OPERATING CONDITIONS

Item	Symbol	min	typ	max	Unit
Supply Voltage	V_{CC}*	4.75	5.0	5.25	V
Input Voltage	V_{IL}*	-0.3	–	0.8	V
	V_{IH}*	2.0	–	V_{CC}	
Operating Temperature	T_{opr}	-20	25	75	°C

* With respect to V_{SS} (SYSTEM GND)

■ ELECTRICAL CHARACTERISTICS

• DC CHARACTERISTICS (V_{CC}=5.0V±5%, V_{SS}=0V, Ta=-20~+75°C, unless otherwise noted.)

Item		Symbol	Test Condition	min	typ*	max	Unit
Input "High" Voltage	All Inputs	V_{IH}		2.0	–	V_{CC}	V
Input "Low" Voltage	All Inputs	V_{IL}		-0.3	–	0.8	V
Input Leakage Current	R/$\overline{W}$, $\overline{RES}$, RS_0, RS_1, CS_0, CS_1, $\overline{CS_2}$, CA_1, CB_1, E	I_{in}	V_{in} = 0~5.25V	-2.5	–	2.5	μA
Three-State (Off State) Input Current	D_0~D_7, PB_0~PB_7, CB_2	I_{TSI}	V_{in} = 0.4~2.4V	-10	–	10	μA
Input "High" Current	PA_0~PA_7, CA_2	I_{IH}	V_{IH} = 2.4V	-200	–	–	μA
Input "Low" Current	PA_0~PA_7, CA_2	I_{IL}	V_{IL} = 0.4V	–	–	-2.4	mA
Output "High" Voltage	D_0~D_7	V_{OH}	I_{OH} = -205μA	2.4	–	–	V
	PA_0~PA_7, CA_2		I_{OH} = -200μA	2.4**	–	–	
			I_{OH} = -10μA	V_{CC}-1.0	–	–	
	PB_0~PB_7, CB_2		I_{OH} = -200μA	2.4	–	–	
Output "Low" Voltage	D_0~D_7, $\overline{IRQA}$, $\overline{IRQB}$	V_{OL}	I_{OL} = 1.6mA	–	–	0.4	V
	Other Outputs		I_{OL} = 1.6mA	–	–	0.4	
			I_{OL} = 3.2mA	–	–	0.6	
Output "High" Current	D_0~D_7	I_{OH}	V_{OH} = 2.4V	-205	–	–	μA
	PA_0~PA_7, CA_2		V_{OH} = 2.4V**	-200	–	–	μA
	PB_0~PB_7, CB_2		V_{OH} = 1.5V	-1.0	–	-10	mA
Output Leakage Current (Off State)	$\overline{IRQA}$, $\overline{IRQB}$	I_{LOH}	V_{OH} = 2.4V	–	–	10	μA
Power Dissipation		P_D		–	260	550	mW
Input Capacitance	PA_0~PA_7, PB_0~PB_7, CA_2, CB_2, D_0~D_7	C_{in}	V_{in} = 0V, Ta = 25°C, f = 1.0MHz	–	–	12.5	pF
	R/$\overline{W}$, $\overline{RES}$, RS_0, RS_1, CS_0, CS_1, $\overline{CS_2}$, CA_1, CB_1, E			–	–	10	
Output Capacitance	$\overline{IRQA}$, $\overline{IRQB}$	C_{out}	V_{in} = 0V, Ta = 25°C, f = 1.0MHz	–	–	10	pF

* Ta = 25°C, V_{CC} = 5.0V ** HD68B21; V_{OH} = 2.2V min (PA_0~PA_7, CA_2)

● AC CHARACTERISTICS (V_{CC}=5.0V±5%, V_{SS}=0, Ta=-20~+75°C, unless otherwise noted.)

1. PERIPHERAL TIMING

Item		Symbol	Test Condition	HD6821		HD68A21		HD68B21		Unit
				min	max	min	max	min	max	
Peripheral Data Setup Time		t_{PDSU}	Fig. 1	200	–	135	–	100	–	ns
Peripheral Data Hold Time		t_{PDH}	Fig. 1	0	–	0	–	0	–	ns
Delay Time, Enable negative transition to CA_2 negative transition	Enable → CA_2 Negative	t_{CA2}	Fig. 2, Fig. 3	–	1.0	–	0.67	–	0.5	μs
Delay Time, Enable negative transition to CA_2 positive transition	Enable → CA_2 Positive	t_{RS1}	Fig. 2	–	1.0	–	0.67	–	0.5	μs
Rise and Fall Times for CA_1 and CA_2 input signals	CA_1, CA_2	t_r, t_f	Fig. 3	–	1.0	–	1.0	–	1.0	μs
Delay Time from CA_1 active transition to CA_2 positive transition	CA_1 → CA_2	t_{RS2}	Fig. 3	–	2.0	–	1.35	–	1.0	μs
Delay Time, Enable negative transition to Peripheral Data Valid	Enable→Peripheral Data	t_{PDW}	Fig. 4, Fig. 5	–	1.0	–	0.67	–	0.5	μs
Delay Time, Enable negative transition to Peripheral CMOS Data Valid	Enable → Peripheral Data PA_0~PA_7, CA_2	t_{CMOS}	V_{CC} – 30% V_{CC} Fig. 4	–	2.0	–	1.35	–	1.0	μs
Delay Time, Enable positive transition to CB_2 negative position	Enable → CB_2	t_{CB2}	Fig. 6, Fig. 7	–	1.0	–	0.67	–	0.5	μs
Delay Time, Peripheral Data Valid to CB_2 negative transition	Peripheral Data → CB_2	t_{DC}	Fig. 5	20	–	20	–	20	–	ns
Delay Time, Enable positive transition to CB_2 positive transition	Enable → CB_2	t_{RS1}	Fig. 6	–	1.0	–	0.67	–	0.5	μs
Peripheral Control Output Pulse Width, CA_2/CB_2	CA_2, CB_2	PW_{CT}	Fig. 2, Fig. 6	550	–	550	–	500	–	ns
Rise and Fall Time for CB_1 and CB_2 input signals	CB_1, CB_2	t_r, t_f	Fig. 7	–	1.0	–	1.0	–	1.0	μs
Delay Time, CB_1 active transition to CB_2 positive transition	CB_1 → CB_2	t_{RS2}	Fig. 7	–	2.0	–	1.35	–	1.0	μs
Interrupt Release Time, $\overline{IRQA}$ and $\overline{IRQB}$	$\overline{IRQA}$, $\overline{IRQB}$	t_{IR}	Fig. 9	–	1.6	–	1.1	–	0.85	μs
Interrupt Response Time	$\overline{IRQA}$, $\overline{IRQB}$	t_{RS3}	Fig. 8	–	1.0	–	1.0	–	1.0	μs
Interrupt Input Pulse Width	CA_1, CA_2, CB_1, CB_2	PWI	Fig. 8	500**	–	500**	–	500**	–	ns
Reset "Low" Time	$\overline{RES}$*	t_{RL}	Fig. 10	1.0	–	0.66	–	0.5	–	μs

* The Reset line must be "High" a minimum of 1.0μs before addressing the PIA.
** At least one Enable "High" pulse should be included in this period.

2. BUS TIMING

1) READ

Item		Symbol	Test Condition	HD6821		HD68A21		HD68B21		Unit
				min	max	min	max	min	max	
Enable Cycle Time		t_{cycE}	Fig. 11	1000	–	666	–	500	–	ns
Enable Pulse Width, "High"		PW_{EH}	Fig. 11	450	–	280	–	220	–	ns
Enable Pulse Width, "Low"		PW_{EL}	Fig. 11	430	–	280	–	210	–	ns
Enable Pulse Rise and Fall Times		t_{Er}, t_{Ef}	Fig. 11	–	25	–	25	–	25	ns
Setup Time	Address, R/$\overline{W}$–Enable	t_{AS}	Fig. 12	140	–	140	–	70	–	ns
Address Hold Time		t_{AH}	Fig. 12	10	–	10	–	10	–	ns
Data Delay Time		t_{DDR}	Fig. 12	–	320	–	220	–	180	ns
Data Hold Time		t_{DHR}	Fig. 12	10	–	10	–	10	–	ns

2) WRITE

Item		Symbol	Test Condition	HD6821		HD68A21		HD68B21		Unit
				min	max	min	max	min	max	
Enable Cycle Time		t_{cycE}	Fig. 11	1000	–	666	–	500	–	ns
Enable Pulse Width, "High"		PW_{EH}	Fig. 11	450	–	280	–	220	–	ns
Enable Pulse Width, "Low"		PW_{EL}	Fig. 11	430	–	280	–	210	–	ns
Enable Pulse Rise and Fall Times		t_{Er}, t_{Ef}	Fig. 11	–	25	–	25	–	25	ns
Setup Time		t_{AS}	Fig. 13	140	–	140	–	70	–	ns
Address Hold Time	Address, R/$\overline{W}$–Enable	t_{AH}	Fig. 13	10	–	10	–	10	–	ns
Data Setup Time		t_{DSW}	Fig. 13	195	–	80	–	60	–	ns
Data Hold Time		t_{DHW}	Fig. 13	10	–	10	–	10	–	ns

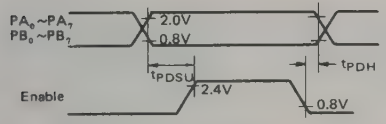
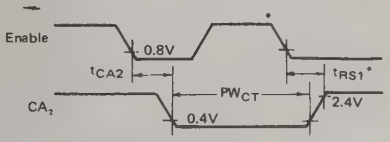


Figure 1 Peripheral Data Setup and Hold Times (Read Mode)



* Assumes part was deselected during the previous E pulse.

Figure 2 CA_2 Delay Time
(Read Mode; CRA5=CRA3=1, CRA4=0)

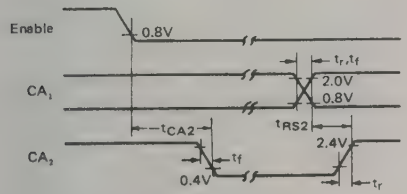


Figure 3 CA_2 Delay Time
(Read Mode; CRA5=1, CRA3=CRA4=0)

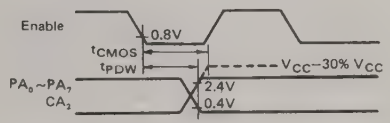
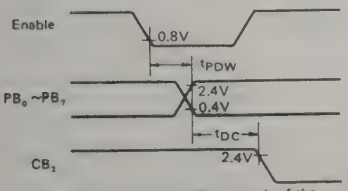
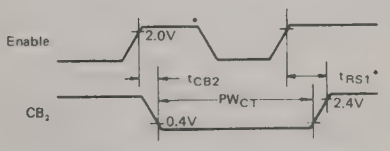


Figure 4 Peripheral CMOS Data Delay Times
(Write Mode; CRA5=CRA3=1, CRA4=0)



(Note) CB_2 goes "Low" as a result of the positive transition of Enable.

Figure 5 Peripheral Data and CB_2 Delay Times
(Write Mode; CRB5=CRB3=1, CRB4=0)



* Assumes part was deselected during the previous E pulse.

Figure 6 CB_2 Delay Time
(Write Mode; CRB5=CRB3=1, CRB4=0)

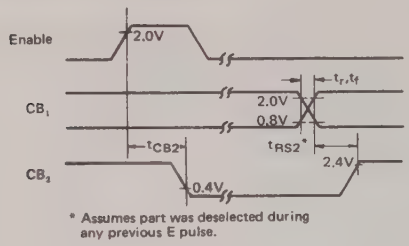


Figure 7 CB_2 Delay Time
(Write Mode; CRB5=1, CRB3=CRB4=0)

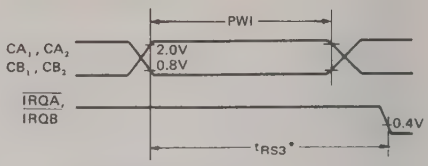


Figure 8 Interrupt Pulse Width and $\overline{IRQ}$ Response

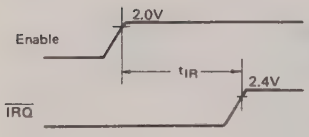


Figure 9 $\overline{IRQ}$ Release Time

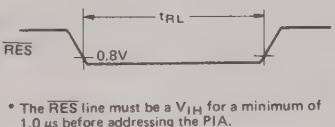


Figure 10 $\overline{RES}$ Low Time

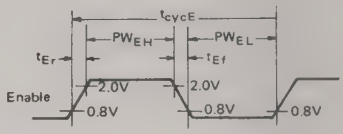


Figure 11 Enable Signal Characteristics

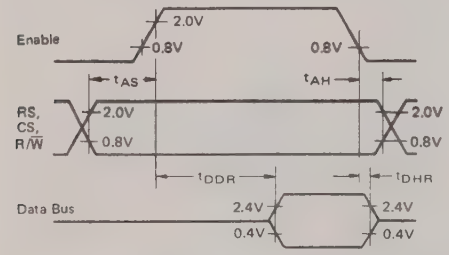


Figure 12 Bus Read Timing Characteristics
(Read Information from PIA)

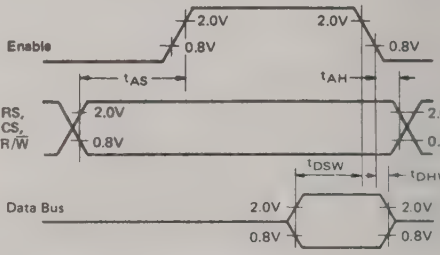


Figure 13 Bus Write Timing Characteristics
(Write Information into PIA)

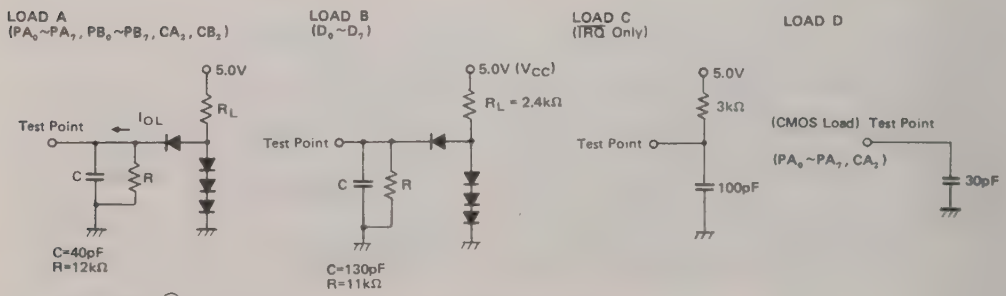


Figure 14 Bus Timing Test Loads

■ PIA INTERFACE SIGNALS FOR MPU

The PIA interfaces to the HD6800 MPU with an eight-bit bi-directional data bus, three chip select lines, two register select lines, two interrupt request lines, read/write line, enable line and reset line. These signals, in conjunction with the HD6800 VMA output, permit the MPU to have complete control over the PIA. VMA should be utilized in conjunction with an MPU address line into a chip select of the PIA.

- **PIA Bi-Directional Data (D_0~D_7)**

The bi-directional data lines ($D_0 \sim D_7$) allow the transfer of data between the MPU and the PIA. The data bus output drivers are three-state devices that remain in the high-impedance (off) state except when the MPU performs a PIA read operation. The R/$\overline{W}$ line is in the Read ("High") state when the PIA is selected for a Read operation.

- **PIA Enable (E)**

The enable pulse, E, is the only timing signal that is supplied to the PIA. Timing of all other signals is referenced to the leading and trailing edges of the E pulse. This signal will normally be a derivative of the HMCS6800 System ϕ_2 Clock. This signal must be continuous clock pulse.

- **PIA Read/Write (R/$\overline{W}$)**

This signal is generated by the MPU to control the direction of data transfers on the Data Bus. A "Low" state on the PIA line enables the input buffers and data is transferred from the MPU to the PIA on the E signal if the device has been selected. A "High" on the R/$\overline{W}$ line sets up the PIA for a transfer of data to the bus. The PIA output buffers are enabled when the proper address and the enable pulse E are present.

- **Reset ($\overline{RES}$)**

The active "Low" $\overline{RES}$ line is used to reset all register bits in the PIA to a logical zero "Low". This line can be used as a power-on reset and as a master reset during system operation.

- **PIA Chip Select (CS_0, CS_1 and $\overline{CS_2}$)**

These three input signals are used to select the PIA. CS_0 and CS_1 must be "High" and $\overline{CS_2}$ must be "Low" for selection of the device. Data transfers are then performed under the control of the E and R/$\overline{W}$ signals. The chip select lines must be stable for the duration of the E pulse. The device is deselected when any of the chip selects are in the inactive state.

- **PIA Register Select (RS_0 and RS_1)**

The two register select lines are used to select the various registers inside the PIA. These two lines are used in conjunction with internal Control Registers to select a particular register that is to be written or read.

The register and chip select lines should be stable for the duration of the E pulse while in the read or write cycle.

- **Interrupt Request ($\overline{IRQA}$ and $\overline{IRQB}$)**

The active "Low" Interrupt Request lines ($\overline{IRQA}$ and $\overline{IRQB}$) act to interrupt the MPU either directly or through interrupt priority circuitry. These lines are "open drain" (no load device on the chip). This permits all interrupt request lines to be tied together in a wire-OR configuration.

Each $\overline{IRQ}$ line has two internal interrupt flag bits that can cause the $\overline{IRQ}$ line to go "Low". Each flag bit is associated with a particular peripheral interrupt line. Also four interrupt enable bits are provided in the PIA which may be used to inhibit a particular interrupt from a peripheral device.

Servicing an interrupt by the MPU may be accomplished by a software routine that, on a prioritized basis, sequentially reads and tests the two control registers in each PIA for interrupt flag bits that are set.

The interrupt flags are cleared (zeroed) as a result of an MPU Read Peripheral Data Operation of the corresponding data register. After being cleared, the interrupt flag bit cannot be enabled to be set until the PIA is deselected during an E pulse. The E pulse is used to condition the interrupt control lines (CA_1, CA_2, CB_1, CB_2). When these lines are used as interrupt inputs at least one E pulse must occur from the inactive edge to the active edge of the interrupt input signal to condition the edge sense network. If the interrupt flag has been enabled and the edge sense circuit has been properly conditioned, the interrupt flag will be set on the next active transition of the interrupt input pin.

■ PIA PERIPHERAL INTERFACE LINES

The PIA provides two 8-bit bi-directional data buses and four interrupt/control lines for interfacing to peripheral devices.

- **Section A Peripheral Data (PA_0~PA_7)**

Each of the peripheral data lines can be programmed to act as an input or output. This is accomplished by setting a "1" in the corresponding Data Direction Register bit for those lines which are to be outputs. A "0" in a bit of the Data Direction Register causes the corresponding peripheral data line to act as an input. During an MPU Read Peripheral Data Operation, the data on peripheral lines programmed to act as inputs appears directly on the corresponding MPU Data Bus lines.

The data in Output Register A will appear on the data lines that are programmed to be outputs. A logical "1" written into the register will cause a "High" on the corresponding data line while a "0" results in a "Low". Data in Output Register A may be read by an MPU "Read Peripheral Data A" operation when the corresponding lines are programmed as outputs. This data will be read properly if the voltage on the peripheral data lines is greater than 2.0 volts for a logic "1" output and less than 0.8 volt for a logic "0" output. Loading the output lines such that the voltage on these lines does not reach full voltage causes the data transferred into the MPU on a Read operation to differ from that contained in the respective bit of Output Register A.

- **Section B Peripheral Data (PB_0~PB_7)**

The peripheral data lines in the B Section of the PIA can be programmed to act as either inputs or outputs in a similar manner to PA_0~PA_7. However, the output buffers driving these lines differ from those driving lines PA_0~PA_7. They have three-state capability, allowing them to enter a high impedance state when the peripheral data line is used as a input. In addition, data on the peripheral data lines PB_0~PB_7 will be read properly from those lines programmed as outputs even if the voltages are below 2.0 volts for a "High". As outputs, these lines are compatible with standard TTL and may also be used as a source of up to 2.5 milliampere (typ.) at 1.5 volts to directly drive the base of a transistor switch.

- **Interrupt Input (CA_1 and CB_1)**

Peripheral Input lines CA_1 and CB_1 are input only lines that set the interrupt flags of the control registers. The active transition for these signals is also programmed by the two control registers.

- **Peripheral Control (CA_2)**

The peripheral control line CA_2 can be programmed to act as an interrupt input or as a peripheral control output. As an output, this line is compatible with standard TTL. The function of this signal line is programmed with Control Register A.

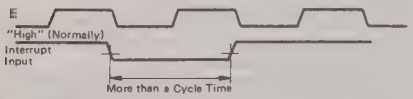
- **Peripheral Control (CB_2)**

Peripheral Control line CB_2 may also be programmed to act as an interrupt input or peripheral control output. As an input,

this line has "High" input impedance and is compatible with standard TTL. As an output it is compatible with standard TTL and may also be used as a source of up to 2.5 milliampere (typ) at 1.5 volts to directly drive the base of a transistor switch. This line is programmed by Control Register B.

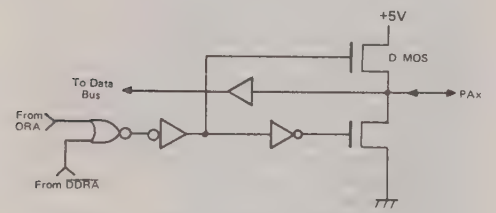
(NOTE) 1. Interrupt inputs CA_1, CA_2, CB_1 and CB_2 shall be used at normal "High" level. When interrupt inputs are "Low" at reset ($\overline{RES}$ = "Low"), interrupt flags CRA6, CRA7, CRB6 and CRB7 may be set.

2. Pulse width of interrupt inputs CA_1, CA_2, CB_1 and CB_2 shall be greater than a E cycle time. In the case that "High" time of E signal is not contained in Interrupt pulse, an interrupt flag may not be set.

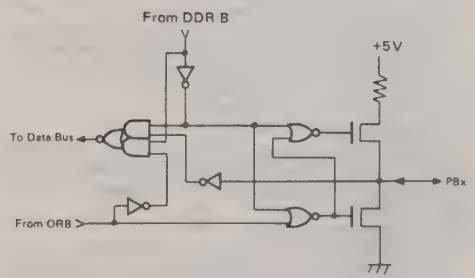


- **The equivalent Circuit of the Lines on Peripheral side**

The equivalent circuit of the lines on Peripheral side is shown in Fig. 15. The output circuits of A port is different from that of B port. When the port is used as input, the input is pullup to V_{CC} side through load MOS in A port and B port becomes "Off" (high impedance).



(a) Section A



(b) Section B

Figure 15 Peripheral Data Bus

■ INTERNAL CONTROLS

There are six locations within the PIA accessible to the MPU data bus: two Peripheral Registers, two Data Direction Registers, and two Control Registers. Selection of these locations is controlled by the RS_0 and RS_1 inputs together with bit 2 in the Control Register, as shown in Table 1.

Table 1 Internal Addressing

RS_1	RS_0	Control Register Bit CRA2	Control Register Bit CRB2	Location Selected
0	0	1	x	Peripheral Register A*
0	0	0	x	Data Direction Register A
0	1	x	x	Control Register A
1	0	x	1	Peripheral Register B*
1	0	x	0	Data Direction Register B
1	1	x	x	Control Register B

x = Don't Care
* Peripheral interface register is a generic term containing peripheral data bus and output register.

- **Initialization**

A "Low" reset line has the effect of zeroing all PIA registers. This will set $PA_0 \sim PA_7$, $PB_0 \sim PB_7$, CA_2 and CB_2 as inputs, and all interrupts disabled. The PIA must be configured during the restart program which follows the reset.

Details of possible configurations of the Data Direction and Control Register are as follows.

- **Data Direction Registers (DDRA and DDRB)**

The two Data Direction Registers allow the MPU to control the direction of data through each corresponding peripheral data line. A Data Direction Register bit set at "0" configures the corresponding peripheral data line as an input; a "1" results in an output.

- **Control Registers (CRA and CRB)**

The two Control Registers (CRA and CRB) allow the MPU to control the operation of the four peripheral control lines CA_1, CA_2, CB_1 and CB_2. In addition they allow the MPU to enable the interrupt lines and monitor the status of the interrupt flags. Bits 0 through 5 of the two registers may be written or read by the MPU when the proper chip select and register select signals are applied. Bits 6 and 7 of the two registers are read only and are modified by external interrupts occurring on control lines CA_1, CA_2, CB_1 or CB_2. The format of the control words is shown in Table 2.

Table 2 Control Word Format

	7	6	5	4	3	2	1	0
CRA	IRQA1	IRQA2	CA_2 Control			DDRA Access	CA_1 Control	
CRB	IRQB1	IRQB2	CB_2 Control			DDRB Access	CB_1 Control	

Data Direction Access Control Bit (CRA2 and CRB2)

Bit 2 in each Control register (CRA and CRB) allows selection of either a Peripheral Interface Register or the Data Direction Register when the proper register select signals are applied to RS_0 and RS_1.

Interrupt Flags (CRA6, CRA7, CRB6, and CRB7)

The four interrupt flag bits are set by active transitions of signals on the four Interrupt and Peripheral Control lines when those lines are programmed to be inputs. These bits cannot be set directly from the MPU Data Bus and are reset indirectly by a Read Peripheral Data Operation on the appropriate section.

Control of CA_1 and CB_1 Interrupt Lines (CRA0, CRB0, CRA1, and CRB1)

The two lowest order bits of the control registers are used to control the interrupt input lines CA_1 and CB_1. Bits CRA0 and CRB0 are used to enable the MPU interrupt signals $\overline{IRQA}$ and $\overline{IRQB}$, respectively. Bits CRA1 and CRB1 determine the active transition of the interrupt input signals CA_1 and CB_1 (Table 3)

Control of CA_2 and CB_2 Peripheral Control Lines (CRA3, CRA4, CRA5, CRB3, CRB4, and CRB5)

Bits 3, 4 and 5 of the two control registers are used to control the CA_2 and CB_2 Peripheral Control lines. These bits determine if the control lines will be an interrupt input or an output control signal. If bit CRA5 (CRB5) is "0" CA_2 (CB_2) is an interrupt input line similar to CA1 (CB1) (Table 4). When CRA5 (CRB5) is "1", CA_2 (CB_2) becomes an output signal that may be used to control peripheral data transfers. When in the output mode, CA_2 and CB_2 have slightly different characteristics (Table 5 and 6).

Table 3 Control of Interrupt Inputs CA_1 and CB_1

CRA1 (CRB1)	CRA0 (CRB0)	Interrupt Input CA_1 (CB_1)	Interrupt Flag CRA7 (CRB7)	MPU Interrupt Request $\overline{IRQA}$ ($\overline{IRQB}$)
0	0	↓ Active	Set "1" on ↓ of CA_1 (CB_1)	Disabled — $\overline{IRQ}$ remains "High"
0	1	↓ Active	Set "1" on ↓ of CA_1 (CB_1)	Goes "Low" when the interrupt flag bit CRA7 (CRB7) goes "1"
1	0	↑ Active	Set "1" on ↑ of CA_1 (CB_1)	Disabled — $\overline{IRQ}$ remains "High"
1	1	↑ Active	Set "1" on ↑ of CA_1 (CB_1)	Goes "Low" when the interrupt flag bit CRA7 (CRB7) goes "1"

(Notes)
1. ↑ indicates positive transition ("Low" to "High")
2. ↓ indicates negative transition ("High" to "Low")
3. The Interrupt flag bit CRA7 is cleared by an MPU Read of the A Peripheral Register and CRB7 is cleared by an MPU Read of the B Peripheral Register.
4. If CRA0 (CRB0) is "0" when an interrupt occurs (Interrupt disabled) and is later brought "1", $\overline{IRQA}$ ($\overline{IRQB}$) occurs after CRA0 (CRB0) is written to a "1".

Table 4 Control of CA_2 and CB_2 as Interrupt Inputs — CRA5 (CRB5) is "0"

CRA5 (CRB5)	CRA4 (CRB4)	CRA3 (CRB3)	Interrupt Input CA_2 (CB_2)	Interrupt Flag CRA6 (CRB6)	MPU Interrupt Request $\overline{IRQA}$ ($\overline{IRQB}$)
0	0	0	↓ Active	Set "1" on ↓ of CA_2 (CB_2)	Disabled — $\overline{IRQ}$ remains "High"
0	0	1	↓ Active	Set "1" on ↓ of CA_2 (CB_2)	Goes "Low" when the interrupt flag bit CRA6 (CRB6) goes "1"
0	1	0	↑ Active	Set "1" on ↑ of CA_2 (CB_2)	Disabled — $\overline{IRQ}$ remains "High"
0	1	1	↑ Active	Set "1" on ↑ of CA_2 (CB_2)	Goes "Low" when the interrupt flag bit CRA6 (CRB6) goes "1"

(Notes)
1. ↑ indicates positive transition ("Low" to "High")
2. ↓ indicates negative transition ("High" to "Low")
3. The interrupt flag bit CRA6 is cleared by an MPU Read of the A Peripheral Register and CRB6 is cleared by an MPU Read of the B Peripheral Register.
4. If CRA3 (CRB3) is "0" when an interrupt occurs (Interrupt disabled) and is later brought "1", $\overline{IRQA}$ ($\overline{IRQB}$) occurs after CRA3 (CRB3) is written to a "1".

Table 5 Control of CB_2 as an Output — CRB5 is "1"

CRB5	CRB4	CRB3	CB_2	
			Cleared	Set
1	0	0	"Low" on the positive transition of the first E pulse after MPU Write "B" Data Register operation.	"High" when the interrupt flag bit CRB7 is set by an active transition of the CB_1 signal. (See Figure 16)
1	0	1	"Low" on the positive transition of the first E pulse after an MPU Write "B" Data Register operation.	"High" on the positive edge of the first "E" pulse following an "E" pulse which occurred while the part was deselected. (See Figure 16)
1	1	0	"Low" (The content of CRB3 is output on CB_2)	
1	1	1	"High" (The content of CRB3 is output on CB_2)	

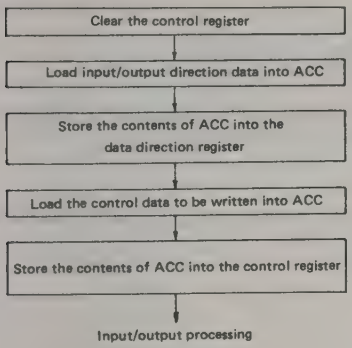
Table 6 Control of CA_2 as an Output — CRA5 is "1"

CRA5	CRA4	CRA3	CA_2	
			Cleared	Set
1	0	0	"Low" on negative transition of E after an MPU Read "A" Data Operation.	"High" when the interrupt flag bit CRA7 is set by an active transition of the CA_1 signal. (See Figure 16)
1	0	1	"Low" on negative transition of E after an MPU Read "A" Data operation.	"High" on the negative edge of the first "E" pulse which occurs during a deselect. (See Figure 16)
1	1	0	"Low" (The content of CRA3 is output on CA_2)	
1	1	1	"High" (The content of CRA3 is output on CA_2)	

■ PIA OPERATION

• Initialization

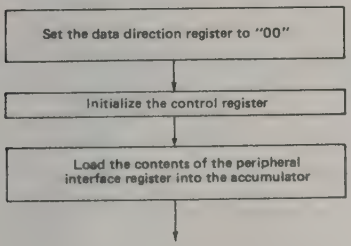
When the external reset input $\overline{RES}$ goes "Low", all internal registers are cleared to "0". Periperal data port (PA_0~PA7, PB_0~PB_7) is defined to be input and control lines (CA_1, CA_2, CB_1 and CB_2) are defined to be the interrupt input lines. PIA is also initialized by software sequence as follows.


Clear the control register
Load input/output direction data into ACC
Store the contents of ACC into the data direction register
Load the control data to be written into ACC
Store the contents of ACC into the control register
Input/output processing

- Program the data direction register access bit of the control register to "0" to allow to access the dada direction register.
- The data of the control line function is set into the accumulator, of which Data Direction Register Access Bit shall be programmed to "1".
- Transfer the control data from the accumulator into the control register.

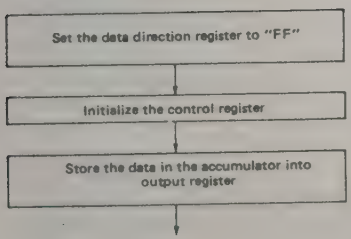
• Read/Write Operation Not Using Control Lines

<Read Operation>


Set the data direction register to "00"
Initialize the control register
Load the contents of the peripheral interface register into the accumulator

```
CLR   CRA     • Clear the DDRA access bit of the control register to "0".
CLR   DDRA    • Clear all bits of the dada direction register.
LDAA  #$04    • Set DDRA access bit of the control register to "1" to allow
STAA  CRA       to access the peripheral interface register.
        :
        :
LDAA  PIRA
```

<Write Operation>


Set the data direction register to "FF"
Initialize the control register
Store the data in the accumulator into output register

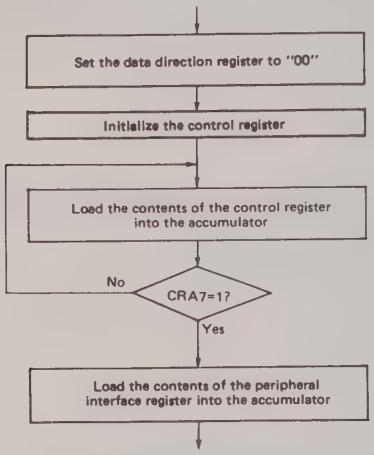
```
CLR   CRA     • Set DDRB access bit of the control register to "0".
LDAA  #$FF    } • Set all bits of the data direction register to "FF".
STAA  DDRB    }

LDAA  #$04    } • Set DDRB access bit of the control register to "1" to allow to
STAA  CRB     }   access the peripheral interface register.
        :
        :
LDAA  DATA    } • Write the data into the peripheral interface register.
STAA  PIRB    }
```

- **Read/Write Operating Using Control Lines**

Read/write request from peripherals shall be put into the control lines as an interrupt signal, and then MPU reads or writes after detecting interrupt request.

< Read >

The following case is that Port A is used and that the rising edge of CA_1 indicates the request for read from peripherals.


Set the data direction register to "00"
Initialize the control register
Load the contents of the control register into the accumulator
No
CRA7=1?
Yes
Load the contents of the peripheral interface register into the accumulator

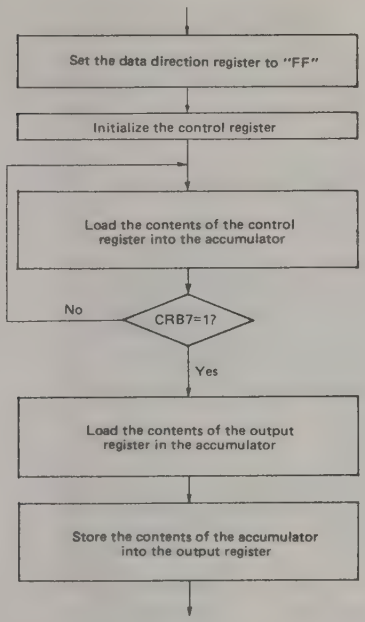
```
CLR   CRA    • Set the DDRA access bit to "0".
CLR   DDRA   • Set all bits of the data direction register to "0".

LDAA  #$06   • Program the rising edge of CA1 to be active. IRQA is masked
STAA  CRA      and DDRA access bit is set to 1.

LOOP  LDAA CRA
      BPL  LOOP   • Check whether the read request comes from peripherals
                    or not.

LDAA  PIRA   • Load the data from the peripheral interface register into the
               accumulator. CRA flag is reset after this read operation.
```

To read the peripheral data, the data is directly transfered to the data buses D_0~D_7 through PA_0~PA_7 or PB_0~PB_7 and they are not latched in the PIA. If necessary, the data should be held in the external latch until MPU completes reading it.

When initializing the control register, interrupt flag bit (CRA7, CRA6, CRB7, CRB6) cannot be written from MPU. If necessary the interrupt flag must be reset by dummy read of Peripheral Register A and B.

<Write>

Write operation using the interrupt signal is as follows. In this case, B port is used and interrupt request is input to CB_1. And the $\overline{IRQ}$ flag is set at the rising edge of CB_1.


Set the data direction register to "FF"
Initialize the control register
Load the contents of the control register into the accumulator
No
CRB7=1?
Yes
Load the contents of the output register in the accumulator
Store the contents of the accumulator into the output register

```
CLR   CRB
LDAA  #$FF
STAA  DDRB
```
• Set DDRB access bit to "0"
• Set all bits of DDRB to output "1".

```
LDAA  #$06
STAA  CRB
```
• Program the rising edge of CB_1 to be active. $\overline{IRQB}$ is masked and DDRB access bit is set to "1".

```
LOOP  LDAA  CRB
      BPL   LOOP
```
• Check whether the write request comes from peripherals or not.

```
LDAA  PIRB
```
• Reset the CRB7 flag by the dummy read of the peripheral interface register.

```
STAB  PIRB
```
• Store the data of the accumulator B to the peripheral interface register.

Interrupt request flag bits (CRA7, CRA6, CRB7 and CRB6) cannot be written and they cannot be also reset by write operation to the peripheral interface register. So dummy read of peripheral interface register is needed to reset the flags.

To accept the next interrupt, it is essential to reset indirectly the interrupt flag by dummy read of peripheral interface register.

Software poling method mentioned above requires MPU to continuously monitor the control register to detect the read/write request from peripherals. So other programs cannot run at the same time. To avoid this problem, hardware interrupt may be used. The MPU is interrupted by $\overline{IRQA}$ or $\overline{IRQB}$ when the read/write request is occurred from peripherals and then MPU analyzes cause of the interrupt request during interrupt processing.

- **Handshake Mode**

The functions of CRA and CRB are similar but not identical in the hand-shake modes. Port A is used for read hand-shake operation and Port B is used for write hand-shake mode.

CA_1 and CB_1 are used for interrupt input requests and CA_2 and CB_2 are control outputs (answer) in hand-shake mode.

Fig. 16, Fig. 17 and Fig. 18 show the timing of hand-shake mode.

<Read Hand-shake Mode>

CRA5="1", CRA4="0" and CRA3="0"

① A peripheral device puts the 8-bit data on the peripheral data lines after the control output CA_2 goes "Low".

② The peripheral requests MPU to read the data by using CA_1 input.

③ CRA7 flag is set and CA_2 becomes "High" (CA_2 automatically becomes "High" by the interrupt CA_1). This indicates the peripheral to maintain the current data and not to transfer the next data.

④ MPU accepts the read request by $\overline{IRQA}$ hardware interrupt or CRA read. Then MPU reads the peripheral register A.

⑤ CA_2 goes "Low" on the following edge of read Enable pulse. This informs that the peripheral can set the next data to port A.

<Write Hand-shake>

CRB5 = "1", CRB4 = "0" and CRB3 = "0"

① A peripheral device requests MPU to write the data by using CB_1 input. CB_2 output remains "High" until MPU write data to the peripheral interface register.

② CRB7 flag is set and MPU accepts the write request.

③ MPU reads the peripheral interface register to reset CRB7 (dummy read).

④ Then MPU write data to the peripheral interface register. The data is output to port B through the output register.

⑤ CB_2 automatically becomes "Low" to tell the peripheral that new data is on port B.

⑥ The peripheral read the data on Port B peripheral data lines and set CB_1 to "Low" to tell MPU that the data on the peripheral data lines has been taken and that next data can be written to the peripheral interface register.

<Pulse mode>

CRA5 = "1", CRA4 = "0" and CRA3 = "1"
CRB5 = "1", CRB4 = "0" and CRB3 = "1"
This mode is shown in Figure 16, Figure 19 and Figure 20.

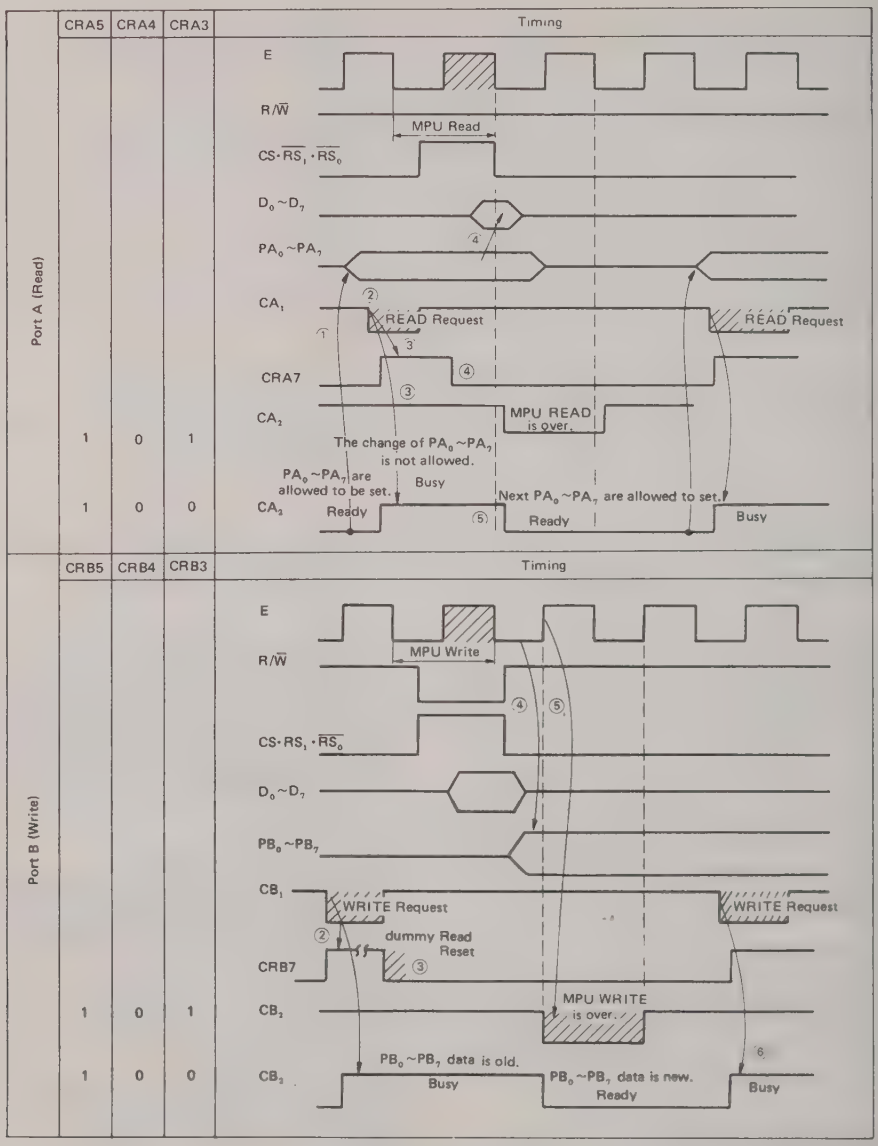


Figure 16 Timing of Hand-shake Mode and Pulse Mode

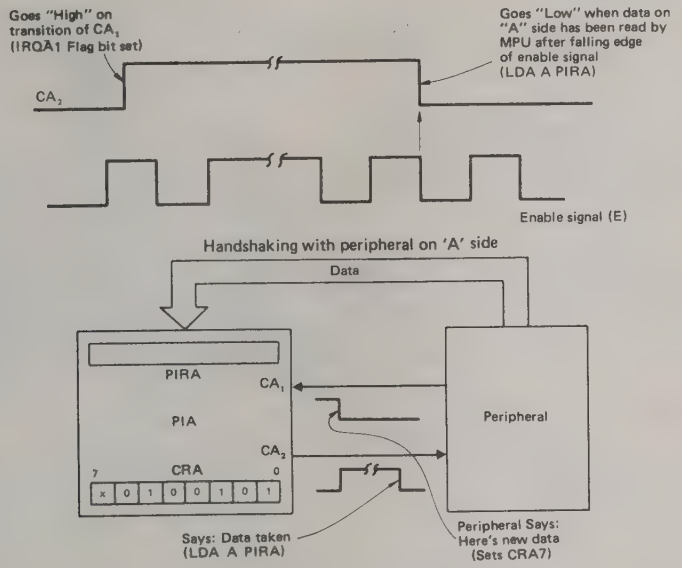

Goes "High" on transition of CA1 (IRQA1 Flag bit set)
Goes "Low" when data on "A" side has been read by MPU after falling edge of enable signal (LDA A PIRA)
CA2
Enable signal (E)
Handshaking with peripheral on 'A' side
Data
PIRA
CA1
PIA
Peripheral
CA2
7
CRA
0
x 0 1 0 0 1 0 1
Says: Data taken (LDA A PIRA)
Peripheral Says: Here's new data (Sets CRA7)

Figure 17 Bits 5, 4, 3 of CRA = 100 (Hand-shake Mode)

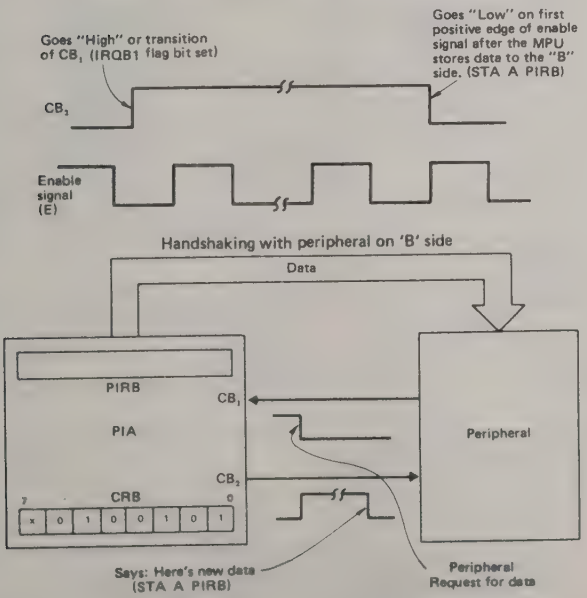

Goes "High" or transition of CB1 (IRQB1 flag bit set)
Goes "Low" on first positive edge of enable signal after the MPU stores data to the "B" side. (STA A PIRB)
CB2
Enable signal (E)
Handshaking with peripheral on 'B' side
Data
PIRB
CB1
PIA
Peripheral
CB2
7
CRB
0
x 0 1 0 0 1 0 1
Says: Here's new data (STA A PIRB)
Peripheral Request for data

Figure 18 Bits 5, 4, 3 of CRB = 100 (Hand-shake Mode)

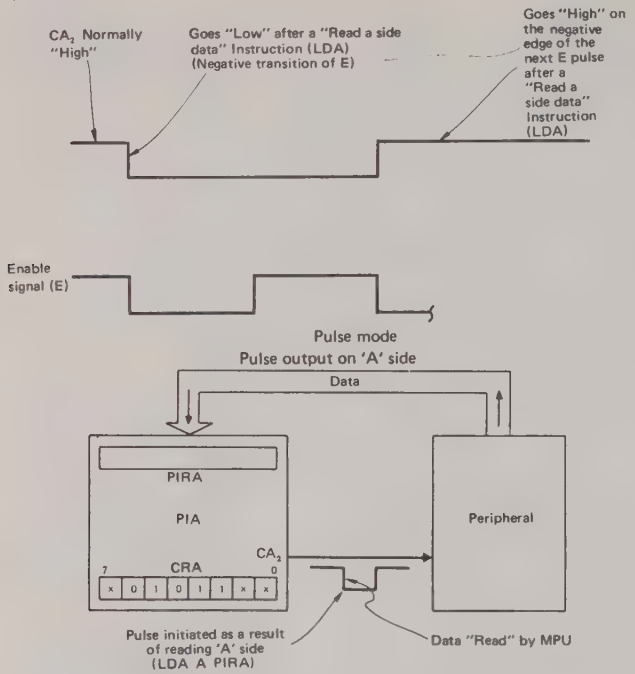

CA2 Normally "High"
Goes "Low" after a "Read a side data" Instruction (LDA) (Negative transition of E)
Goes "High" on the negative edge of the next E pulse after a "Read a side data" Instruction (LDA)
Enable signal (E)
Pulse mode
Pulse output on 'A' side
Data
PIRA
PIA
CRA
CA2
Peripheral
Pulse initiated as a result of reading 'A' side (LDA A PIRA)
Data "Read" by MPU

Figure 19 Bits 5, 4, 3 of CRA = 101 (Pulse Mode)

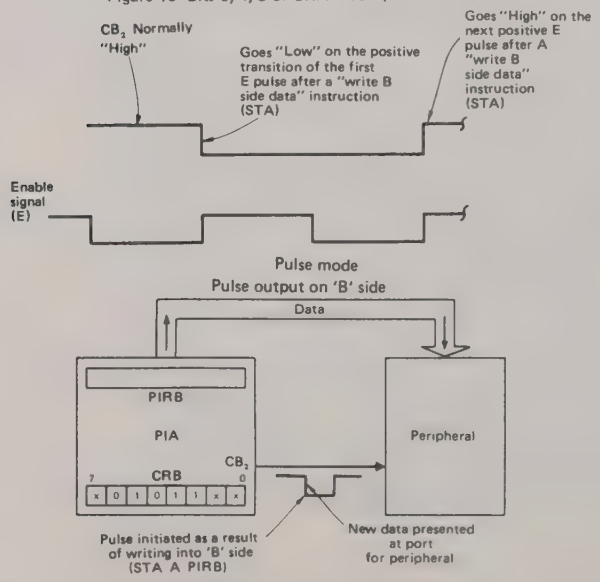

CB2 Normally "High"
Goes "Low" on the positive transition of the first E pulse after a "write B side data" instruction (STA)
Goes "High" on the next positive E pulse after A "write B side data" instruction (STA)
Enable signal (E)
Pulse mode
Pulse output on 'B' side
Data
PIRB
PIA
CRB
CB2
Peripheral
Pulse initiated as a result of writing into 'B' side (STA A PIRB)
New data presented at port for peripheral

Figure 20 Bits 5, 4, 3 of CRB=101 (Pulse Mode)

■ SUMMARY OF CONTROL REGISTERS CRA AND CRB

Control registers CRA and CRB have total control of CA_1, CA_2, CB_1, and CB_2 lines. The status of eight bits of the control registers may be read into the MPU. However, the MPU can only write into Bit 0 through Bit 5 (6 bits), since Bit 6 and Bit 7 are set only by CA_1, CA_2, CB_1, or CB_2.

● Addressing PIAs

Before addressing PIAs, the data direction (DDR) must first be loaded with the bit pattern that defines how each line is to function, i.e., as an input or an output. A logic "1" in the data direction register defines the corresponding line as an output while a logic "0" defines the corresponding line as an input. Since the DDR and the peripheral interface resister have the same address, the control register bit 2 determines which register is being addressed. If Bit 2 in the control register is a logic "0", then the DDR is addressed. If Bit 2 in the control register is a logic "1", the peripheral interface register is addressed. Therefore, it is essential that the DDR be loaded first before setting Bit 2 of the control register.

<Example>

Given a PIA with an address of 4004, 4005, 4006, and 4007. 4004 is the address of the A side peripheral interface register. 4005 is the address of the A side control register. 4006 is the address of the B side peripheral interface register. 4007 is the address of the B side control register. On the A side, Bits 0, 1, 2, and 3 will be defined as inputs, while Bits 4, 5, 6, and 7 will be used as outputs. On the B side, all lines will be used as outputs.

```
PIA1AD = 4004      (DDRA, PIRA)
PIA1AC = 4005      (CRA)
PIA1BD = 4006      (DDRB, PIRB)
PIA1BC = 4007      (CRB)

1.  LDA A #%11110000   (4 outputs, 4 inputs)
2.  STA A    PIA1AD    (Loads A DDR)
3.  LDA A #%11111111   (All outputs)
4.  STA A    PIA1BD    (Loads B DDR)
5.  LDA A #%00000100   (Sets Bit 2)
6.  STA A    PIA1AC    (Bit 2 set in A control register)
7.  STA A    PIA1BC    (Bit 2 set in B control register)
```

Statement 2 addresses the DDR, since the control register (Bit 2) has not been loaded. Statements 6 and 7 load the control registers with Bit 2 set, so addressing PIA1AD or PIA1BD accesses the peripheral interface register.

● PIA Programming Via The Index Register

The program shown in the previous section can be accomplished using the Index Register.

```
1.  LDX   #$F004
2.  STX   PIA1AD     $F0→PIA1AD;$04→PIA1AC
3.  LDX   #$FF04
4.  STX   PIA1BD     $FF→PIA1BD;$04→PIA1BC
```

Using the index register in this example has saved six bytes of program memory as compared to the program shown in the previous section.

● Active Low Outputs

When all the outputs of given PIA port are to be active "Low" (True ≦ 0.4 volts), the following procedure should be used.

a) Set Bit 2 in the control register.
b) Store all 1s ($FF) in the peripheral interface register.
c) Clear Bit 2 in the control register.
d) Store all 1s ($FF) in the data direction register.
e) Store control word (Bit 2 = 1) in control register.

<Example>

The B side of PIA1 is set up to have all active low outputs. CB_1 and CB_2 are set up to allow interrupts in the HANDSHAKE MODE and CB_1 will respond to positive edges ("Low"-to-"High" transitions). Assume reset conditions. Addresses are set up and equated to the same labels as previous example.

```
1.  LDA A #4
2.  STA A PIA1BC     Set Bit 2 in PIA1BC (control register)
3.  LDA B #$FF
4.  STA B PIA1BD     All 1s in peripheral interface register
5.  CLR   PIA1BC     Clear Bit 2
6.  STA B PIA1BD     All 1s in data direction register
7.  LDA A #$27
8.  STA A PIA1BC     00100111→→ control register
```

The above procedure is required in order to avoid outputs going "Low", to the active "Low" TRUE STATE, when all ls are stored to the data direction register as would be the case if the normal configuration procedure were followed.

● Interchanging RS_0 And RS_1

Some system applications may require movement of 16 bits of data to or from the "outside world" via two PIA ports (A side + B side). When this is the case it is an advantage to interconnect RS_1 and RS_0 as follows.

RS_0 to A1 (Address Line A1)
RS_1 to A0 (Address Line A0)

This will place the peripheral interface registers and control registers side by side in the memory map as follows.

Table	Example Address	
PIA1AD	$4004	(DDRA, PIRA)
PIA1BD	$4005	(DDRB, PIRB)
PIA1AC	$4006	(CRA)
PIA1BC	$4007	(CRB)

The index register or stackpointer may be used to move the 16-bit data in two 8-bit bytes with one instruction. As an example:

LDX PIA1AD PIA1AD →→ IX_H: PIA1BD →→ IX_L

● PIA — After Reset

When the $\overline{RES}$ (Reset Line) has been held "Low" for a minimum of one microsecond, all registers in the PIA will be cleared.

Because of the reset conditions, the PIA has been defined as

follows.

1. All I/O lines to the "outside world" have been defined as inputs.
2. CA_1, CA_2, CB_1, and CB_2 have been defined as interrupt input lines that are negative edge sensitive.
3. All the interrupts on the control lines are masked. Setting of interrupt flag bits will not cause $\overline{IRQA}$ or $\overline{IRQB}$ to go "Low".

■ SUMMARY OF CA_1-CB_1 PROGRAMMING

Bits 1 and 0 of the respective control registers are used to program the interrupt input control lines CA_1 and CB_1.

b1	b0	
0	0	b1 = Edge (0 = –, 1 = +)
0	1	b0 = Mask (0 = Mask, 1 = Allow)
1	0	
1	1	

Note that this is the same logic as Bits 4 and 3 for CA_2-CB_2 when CA_2-CB_2 are programmed as inputs.

■ SUMMARY OF CA_2-CB_2 PROGRAMMING

Bits 5, 4, and 3 of the control registers are used to program the operation of CA_2-CB_2.

	b5	b4	b3		
CA_2–CB_2 Input Mode →	0	0(–)	0	(Mask)	CA_2–CB_2 Input Mode
	0	0(–)	1	(Allow)	b4 = Edge (0 = –, 1 = +)
	0	1(+)	0	(Mask)	b3 = Mask (0 = Mask,
	0	1(+)	1	(Allow)	1 = Allow)
CA_2–CB_2 Output Mode →	1	0			0 – Handshake Mode
	1	0			1 – Pulse Mode
	1	1	0		b3 Following Mode
	1	1	1		

I/O As Follow:
Control Lines:
CA_1 – Positive Edge, Allow Interrupt
CA_2 – Pulse Mode
CB_1 – Negative Edge, Mask Interrupt
CB_2 – Hand Shake Mode

Assume Reset Condition
PIA1AD
PIA1AC
PIA1BD
PIA1BC

PIA Configuration Solution

LDA A #$BC	10111100
STA A PIA1AD	I/O to DDRA
LDA A #$FF	1111 1111
STA A PIA1BD	I/O to DDRB
LDA A #$2F	0010 1111
STA A PIA1AC	To "A" Control
LDA A #$24	0010 0100
STA A PIA1BC	To "B" Control

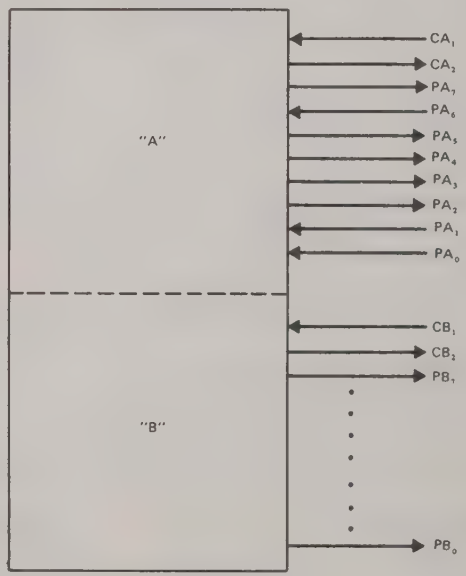


Figure 21 PIA Configuration Problem

HD6321,HD63A21,HD63B21
CMOS PIA (Peripheral Interface Adapter)
—PRELIMINARY—

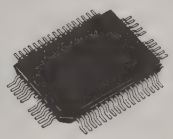
The HD6321 is a CMOS Peripheral Interface Adapter provides the universal means of interfacing peripheral equipment to the HD6800 Microprocessing Unit (MPU). This device is capable of interfacing the MPU to peripherals through two 8-bit bi-directional peripheral data buses and four control lines. No external logic is required for interfacing to most peripheral devices.

The functional configuration of the PIA is programmed by the MPU during system initialization. Each of the peripheral data lines can be programmed to act as an input or output, and each of the four control/interrupt lines may be programmed for one of several control mode. This allows a high degree of flexibility in the over-all operation of the interface. Exceeding Low power dissipation is realized due to adopting CMOS process.

HD6321P,HD63A21P,HD63B21P

(DP-40)

HD6321FP, HD63A21FP, HD63B21FP



(FP-54)

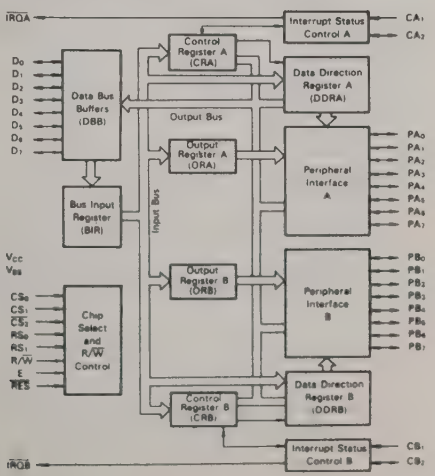
■ **FEATURES**

- Low-Power, High-Speed, High-Density CMOS
- Compatible with NMOS PIA (HD6821) (Refer to Electrical Specification as to Minor difference.)
- Two Bi-directional 8-Bit Peripheral Data Bus for interface to Peripheral devices
- High-Impedance 3-State and Direct Transistor Drive Peripheral Lines
- Two TTL Drive Capability on All A and B Side Buffers
- Handshake Control Logic for Input and Output Peripheral Operation

  CA_1, CA_2 ..... Port A ($PA_0 \sim PA_7$)

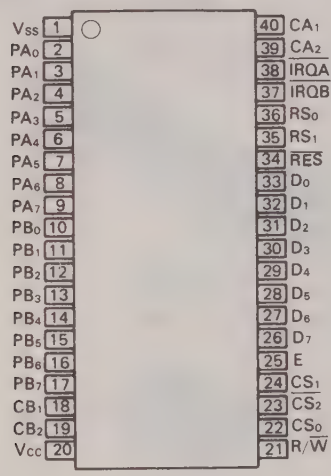
  CB_1, CB_2 ..... Port B ($PB_0 \sim PB_7$)
- Two Programmable Control Registers (CRA, CRB)
- Two Programmable Data Direction Registers (DDRA, DDRB)

■ **BLOCK DIAGRAM**



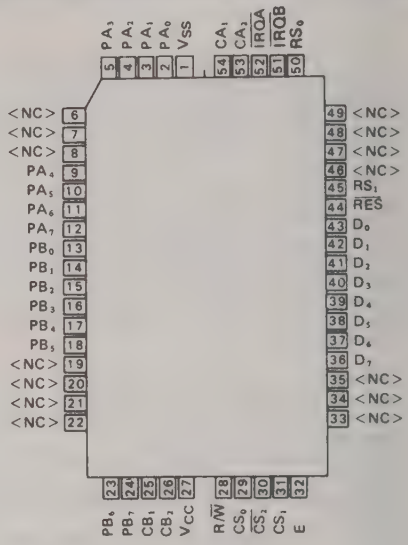
■ PIN ARRANGEMENT

● HD6321P, HD63A21P, HD63B21P



(Top View)

● HD6321FP, HD63A21FP, HD63B21FP



(Top View)

■ ABSOLUTE MAXIMUM RATINGS

Item	Symbol	Value	Unit
Supply Voltage	V_{CC}*	-0.3 ~ +7.0	V
Input Voltage	V_{in}*	-0.3 ~ +7.0	V
Maximum Output Current	$\|I_O\|$**	10	mA
Maximum Total Output Current	$\|\Sigma I_O\|$***	100	mA
Operating Temperature	T_{opr}	-20 ~ +75	°C
Storage Temperature	T_{stg}	-55 ~ +150	°C

* With respect to V_{SS} (SYSTEM GND)

** Maximum output current is the maximum current which can flow in or flow out from one output terminal and I/O common terminal. ($PA_0 \sim PA_7$, CA_2, $PB_0 \sim PB_7$, CB_2, $D_0 \sim D_7$)

*** Maximum total output current is the total sum of output current which can flow in or flow out simultaneously from output terminals and I/O common terminals.($PA_0 \sim PA_7$, CA_2, $PB_0 \sim PB_7$, CB_2, $D_0 \sim D_7$)

(NOTE) Permanent LSI damage may occur if maximum ratings are exceeded. Normal operation should be under recommended operating conditions. If these conditions are exceeded, it could affect reliability of LSI.

■ RECOMMENDED OPERATING CONDITIONS

Item		Symbol	min	typ	max	Unit
Supply Voltage		V_{CC}*	4.5	5.0	5.5	V
Input "Low" Voltage		V_{IL}*	0	–	0.8	V
Input "High" Voltage	$D_0 \sim D_7$, $PA_0 \sim PA_7$, CA_1, CA_2, $PB_0 \sim PB_7$, CB_1, CB_2	V_{IH}*	2.2	–	V_{CC}	V
	E, R/$\overline{W}$, CS_0, $\overline{CS_2}$, CS_1, RS_0, RS_1, $\overline{RES}$		3.0**			
Operating Temperature		T_{opr}	-20	25	75	°C

* With respect to V_{SS} (SYSTEM GND)
** Characteristics will be improved.

■ ELECTRICAL CHARACTERISTICS

● DC CHARACTERISTICS (V_{CC}=5.0V±10%, V_{SS}=0V, Ta=-20~+75°C, unless otherwise noted.)

Item		Symbol	Test Condition	min	typ	max	Unit
Input "High" Voltage	$D_0 \sim D_7$, $PA_0 \sim PA_7$, CA_1, CA_2, $PB_0 \sim PB_7$, CB_1, CB_2	V_{IH}		2.2	–	V_{CC}	V
	E, R/$\overline{W}$, CS_0, $\overline{CS_2}$, CS_1, RS_0, RS_1, $\overline{RES}$			3.0**			
Input "Low" Voltage	All Inputs	V_{IL}		-0.3	–	0.8	V
Input Leakage Current	R/$\overline{W}$, $\overline{RES}$, $\overline{RS_0}$, RS_1, CS_0, CS_1, $\overline{CS_2}$, CA_1, CB_1, E	I_{in}	Vin = 0 ~ Vcc	-2.5	–	2.5	μA
Three-State (Off State) Input Current	$D_0 \sim D_7$, $PB_0 \sim PB_7$, CB_2	I_{TSI}	Vin = 0.4 ~ Vcc	-10	–	10	μA
Output "High" Voltage	$D_0 \sim D_7$	V_{OH}	I_{OH} = -400 μA	4.1	–	–	V
			$I_{OH} \leq$ -10 μA	V_{CC} -0.1	–	–	
	$PA_0 \sim PA_7$, CA_2 $PB_0 \sim PB_7$, CB_2		I_{OH} = -400 μA	4.1	–	–	
			$I_{OH} \leq$ -10 μA	V_{CC} -0.1	–	–	
Output "Low" Voltage	$D_0 \sim D_7$, $\overline{IRQA}$, $\overline{IRQB}$	V_{OL}	I_{OL} = 1.6mA	–	–	0.4	V
	$PA_0 \sim PA_7$, CA_2 $PB_0 \sim PB_7$, CB_2		I_{OL} = 3.2mA	–	–	0.6	
Output Leakage Current (Off State)	$\overline{IRQA}$, $\overline{IRQB}$	I_{LOH}	V_{OH} = V_{CC}	–	–	10	μA
Input Capacitance	$PA_0 \sim PA_7$, $PB_0 \sim PB_7$, CA_2, CB_2, $D_0 \sim D_7$	C_{in}	V_{in} = 0V, Ta = 25°C, f = 1.0MHz	–	–	12.5	pF
	R/$\overline{W}$, $\overline{RES}$, $\overline{RS_0}$, RS_1, CS_0, CS_1, $\overline{CS_2}$, CA_1, CB_1, E			–	–	10	
Output Capacitance	$\overline{IRQA}$, $\overline{IRQB}$	C_{out}	V_{in} = 0V, Ta = 25°C, f = 1.0MHz	–	–	10	pF
Supply Current*	• $PA_0 \sim PA_7$, CA_2 and $PB_0 \sim PB_7$, CB_1 are specified as input. • Chip is not selected • Input level (Except E) V_{IH} min = V_{CC}-0.8V V_{IL} max = 0.8V	I_{cc}	E = 1.0MHz	–	–	300	μA
			E = 1.5MHz	–	–	400	
			E = 2.0MHz	–	–	500	
	• $PA_0 \sim PA_7$, CA_2 and $PB_0 \sim PB_7$, CB_2 are specified as input. • Under Data Bus R/$\overline{W}$ operation		E = 1.0MHz	–	–	4	mA
			E = 1.5MHz	–	–	5	
			E = 2.0MHz	–	–	6	

* Supply current is defined on the condition that there is no current flow from output terminals. Supply current will be increased when the current from output terminal exists. Also the current will be increased for charging and discharging the capacitive load. Please take this case into consideration in estimating system power.
** Characteristics will be improved.

• AC CHARACTERISTICS (V_{CC}=5.0V±10%,V_{SS}=0, Ta=-20~+75°C, unless otherwise noted.)

1. PERIPHERAL TIMING

Item		Symbol	Test Condition	HD6321		HD63A21		HD63B21		Unit
				min	max	min	max	min	max	
Peripheral Data Setup Time		t_{PDSU}	Fig. 1	100	–	100	–	100	–	ns
Peripheral Data Hold Time		t_{PDH}	Fig. 1	0	–	0	–	0	–	ns
Delay Time, Enable negative transition to CA_2 negative transition	Enable → CA_2 Negative	t_{CA2}	Fig. 2, Fig. 3	–	200	–	200	–	200	ns
Delay Time, Enable negative transition to CA_2 positive transition	Enable → CA_2 Positive	t_{RS1}	Fig. 2	–	200	–	200	–	200	ns
Rise and Fall Times for CA_1 and CA_2 input signals	CA_1, CA_2	t_r, t_f	Fig. 3	–	100	–	100	–	100	ns
Delay Time from CA_1 active transition to CA_2 positive transition	CA_1 → CA_2	t_{RS2}	Fig. 3	–	300	–	300	–	300	ns
Delay Time, Enable negative transition to Peripheral Data Valid	Enable→Peripheral Data	t_{PDW}	Fig. 4, Fig. 5	–	300	–	300	–	300	ns
Delay Time, Enable positive transition to CB_2 negative transition	Enable → CB_2	t_{CB2}	Fig. 6, Fig. 7	–	200	–	200	–	200	ns
Delay Time, Peripheral Data Valid to CB_2 negative transition	Peripheral Data → CB_2	t_{DC}	Fig. 5	20	–	20	–	20	–	ns
Delay Time, Enable positive transition to CB_2 positive transition	Enable → CB_2	t_{RS1}	Fig. 6	–	200	–	200	–	200	ns
Peripheral Control Output Pulse Width, CA_2/CB_2	CA_2, CB_2	PW_{CT}	Fig. 2, Fig. 6	550	–	375	–	250	–	ns
Rise and Fall Time for CB_1 and CB_2 input signals	CB_1, CB_2	t_r, t_f	Fig. 7	–	100	–	10(	–	100	ns
Delay Time, CB_1 active transition to CB_2 positive transition	CB_1 → CB_2	t_{RS2}	Fig. 7	–	300	–	300	–	300	ns
Interrupt Release Time, $\overline{IRQA}$ and $\overline{IRQB}$	$\overline{IRQA}$, $\overline{IRQB}$	t_{IR}	Fig. 9	–	800	–	800	–	800	ns
Interrupt Response Time	$\overline{IRQA}$, $\overline{IRQB}$	t_{RS3}	Fig. 8	–	400	–	400	–	400	ns
Interrupt Input Pulse Width	CA_1, CA_2, CB_1, CB_2	PW_I	Fig. 8	1E cycle	–	1E cycle	–	1E cycle	–	
Reset "Low" Time	$\overline{RES}$*	t_{RL}	Fig. 10	200	–	200	–	200	–	ns

* The Reset line must be "High" a minimum of 1.0μs before addressing the PIA.
** At least one Enable "High" pulse should be included in this period.

2. BUS TIMING

1) READ

Item		Symbol	Test Condition	HD6321		HD63A21		HD63B21		Unit
				min	max	min	max	min	max	
Enable Cycle Time		t_{cycE}	Fig. 11	1000	–	666	–	500	–	ns
Enable Pulse Width, "High"		PW_{EH}	Fig. 11	450	–	280	–	220	–	ns
Enable Pulse Width, "Low"		PW_{EL}	Fig. 11	430	–	280	–	210	–	ns
Enable Pulse Rise and Fall Times		t_{Er}, t_{Ef}	Fig. 11	–	25	–	25	–	20	ns
Setup Time	Address, R/$\overline{W}$–Enable	t_{AS}	Fig. 12	80	–	60	–	60*	–	ns
Address Hold Time		t_{AH}	Fig. 12	10	–	10	–	10	–	ns
Data Delay Time		t_{DDR}	Fig. 12	–	290	–	180	–	150	ns
Data Hold Time		t_{DHR}	Fig. 12	20	100	20	100	20	100	ns

* Characteristics will be improved.

2) WRITE

Item		Symbol	Test Condition	HD6321		HD63A21		HD63B21		Unit
				min	max	min	max	min	max	
Enable Cycle Time		t_{cycE}	Fig. 11	1000	–	666	–	500	–	ns
Enable Pulse Width, "High"		PW_{EH}	Fig. 11	450	–	280	–	220	–	ns
Enable Pulse Width, "Low"		PW_{EL}	Fig. 11	430	–	280	–	210	–	ns
Enable Pulse Rise and Fall Times		t_{Er}, t_{Ef}	Fig. 11	–	25	–	25	–	20	ns
Setup Time		t_{AS}	Fig. 13	80	–	60	–	60*	–	ns
Address Hold Time	Address, R/$\overline{W}$–Enable	t_{AH}	Fig. 13	10	–	10	–	10	–	ns
Data Setup Time		t_{DSW}	Fig. 13	165	–	80	–	60	–	ns
Data Hold Time		t_{DHW}	Fig. 13	10	–	10	–	10	–	ns

* Characteristics will be improved.

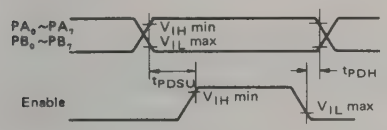
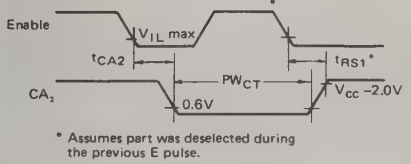


Figure 1 Peripheral Data Setup and Hold Times (Read Mode)



* Assumes part was deselected during the previous E pulse.

Figure 2 CA_2 Delay Time (Read Mode; CRA5=CRA3=1, CRA4=0)

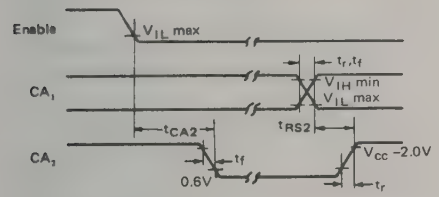


Figure 3 CA_2 Delay Time (Read Mode; CRA5=1, CRA3=CRA4=0)

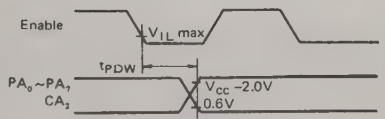
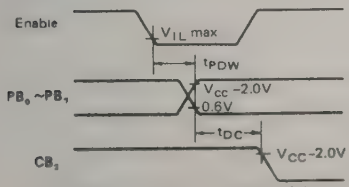
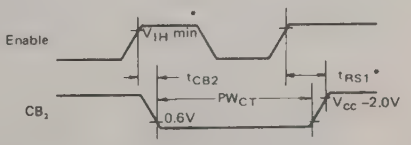


Figure 4 Peripheral Data Delay Times (Write Mode; CRA5=CRA3=1, CRA4=0)



(Note) CB_2 goes "Low" as a result of the positive transition of Enable.

Figure 5 Peripheral Data and CB_2 Delay Times (Write Mode; CRB5=CRB3=1, CRB4=0)



* Assumes part was deselected during the previous E pulse.

Figure 6 CB_2 Delay Time (Write Mode; CRB5=CRB3=1, CRB4=0)

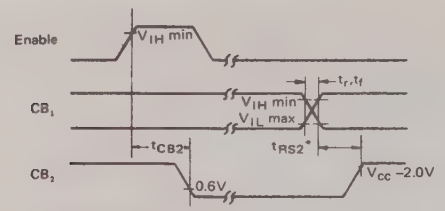


Figure 7 CB_2 Delay Time (Write Mode; CRB5=1, CRB3=CRB4=0)

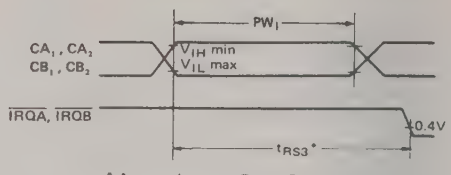


Figure 8 Interrupt Pulse Width and $\overline{IRQ}$ Response

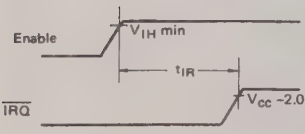


Figure 9 $\overline{IRQ}$ Release Time

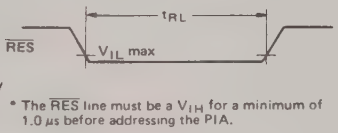


Figure 10 $\overline{RES}$ Low Time

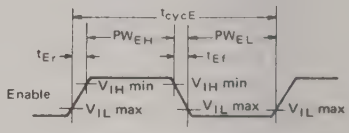


Figure 11 Enable Signal Characteristics

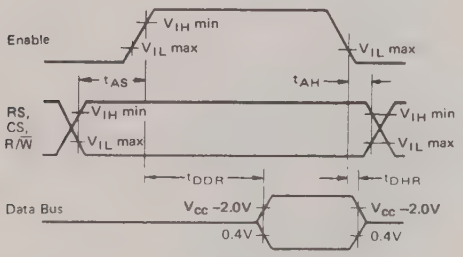


Figure 12 Bus Read Timing Characteristics (Read Information from PIA)

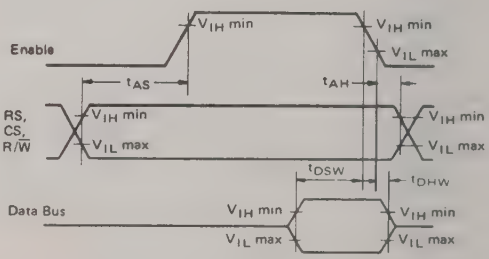


Figure 13 Bus Write Timing Characteristics (Write Information into PIA)

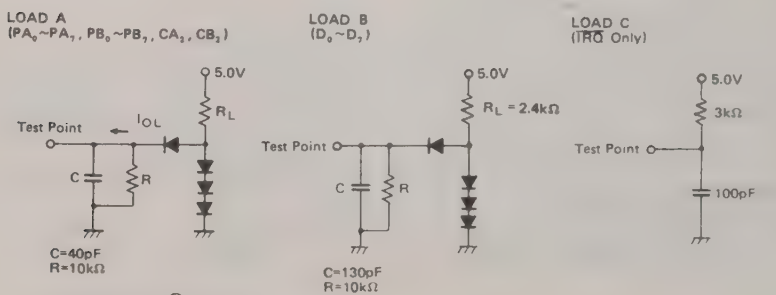


Figure 14 Bus Timing Test Loads

■ PIA INTERFACE SIGNALS FOR MPU

The PIA interfaces to the HD6800 MPU with an eight-bit bi-directional data bus, three chip select lines, two register select lines, two interrupt request lines, read/write line, enable line and reset line. These signals, in conjunction with the HD6800 VMA output, permit the MPU to have complete control over the PIA. VMA should be utilized in conjunction with an MPU address line into a chip select of the PIA.

- **Bi-Directional Data ($D_0 \sim D_7$)**

The bi-directional data lines ($D_0 \sim D_7$) allow the transfer of data between the MPU and the PIA. The data bus output drivers are three-state devices that remain in the high-impedance (off) state except when the MPU performs a PIA read operation. The R/$\overline{W}$ line is in the Read ("High") state when the PIA is selected for a Read operation.

- **Enable (E)**

The enable pulse, E, is the only timing signal that is supplied to the PIA. Timing of all other signals is referenced to the leading and trailing edges of the E pulse. This signal will normally be a derivative of the HMCS6800 System ϕ_2 Clock. This signal must be continuous clock pulse.

- **Read/Write (R/$\overline{W}$)**

This signal is generated by the MPU to control the direction of data transfers on the Data Bus. A "Low" state on the PIA line enables the input buffers and data is transferred from the MPU to the PIA on the E signal if the device has been selected. A "High" on the R/$\overline{W}$ line sets up the PIA for a transfer of data to the bus. The PIA output buffers are enabled when the proper address and the enable pulse E are present.

- **Reset ($\overline{RES}$)**

The active "Low" $\overline{RES}$ line is used to reset all register bits in the PIA to a logical zero "Low". This line can be used as a power-on reset and as a master reset during system operation.

- **Chip Select (CS_0, CS_1 and $\overline{CS_2}$)**

These three input signals are used to select the PIA. CS_0 and CS_1 must be "High" and $\overline{CS_2}$ must be "Low" for selection of the device. Data transfers are then performed under the control of the E and R/$\overline{W}$ signals. The chip select lines must be stable for the duration of the E pulse. The device is deselected when any of the chip selects are in the inactive state.

- **Register Select (RS_0 and RS_1)**

The two register select lines are used to select the various registers inside the PIA. These two lines are used in conjunction with internal Control Registers to select a particular register that is to be written or read.

The register and chip select lines should be stable for the duration of the E pulse while in the read or write cycle.

- **Interrupt Request ($\overline{IRQA}$ and $\overline{IRQB}$)**

The active "Low" Interrupt Request lines ($\overline{IRQA}$ and $\overline{IRQB}$) act to interrupt the MPU either directly or through interrupt priority circuitry. These lines are "open drain" (no load device on the chip). This permits all interrupt request lines to be tied together in a wire-OR configuration.

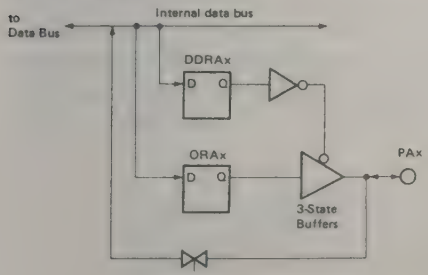
Each $\overline{IRQ}$ line has two internal interrupt flag bits that can cause the $\overline{IRQ}$ line to go "Low ". Each flag bit is associated with a particular peripheral interrupt line. Also four interrupt enable bits are provided in the PIA which may be used to inhibit a particular interrupt from a peripheral device.

Servicing an interrupt by the MPU may be accomplished by a software routine that, on a prioritized basis, sequentially reads and tests the two control registers in each PIA for interrupt flag bits that are set.

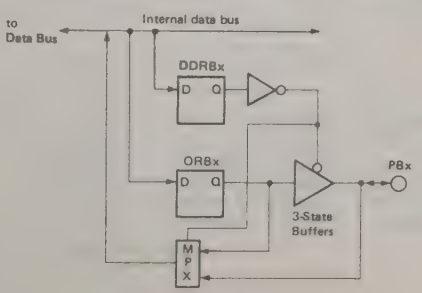
The interrupt flags are cleared (zeroed) as a result of an MPU Read Peripheral Data Operation of the corresponding data register. After being cleared, the interrupt flag bit cannot be enabled to be set until the PIA is deselected during an E pulse. The E pulse is used to condition the interrupt control lines (CA_1, CA_2, CB_1, CB_2). When these lines are used as interrupt inputs at least one E pulse must occur from the inactive edge to the active edge of the interrupt input signal to condition the edge sense network. If the interrupt flag has been enabled and the edge sense circuit has been properly conditioned, the interrupt flag will be set on the next active transition of the interrupt input pin.

■ PIA PERIPHERAL INTERFACE LINES

Port A and Port B provide four interrupt control lines and two sets of 8-bit Bi-directional peripheral data bus for interfacing to input/output divices. Fig. 15 shows the block diagram of Port A and Port B. The output drivers of Port A and Port B consist of three-state drivers, allowing them to enter a High-impedance state when the peripheral data line is used as an input. Port A and Port B have the same output buffer. But the circuit configuration is slightly different and this makes the difference on data flow when MPU reads Port A and Port B in the case each Port is specified as output. As shown in Fig. 15, the output of the peripheral data A is transferred to internal data bus when used as output. On the other hand, in the case of Port B the contents of output register (ORB) is directly transferred to internal data bus through the multiplexor.



(a) Port A



(b) Port B

Figure 15 Block Diagram of Port A and Port B

- **Port A Peripheral Data ($PA_0 \sim PA_7$)**

Each of the peripheral data lines can be programmed to act as an input or output. This is accomplished by setting a "1" in the corresponding Data Direction Register bit for those lines which are to be outputs. A "0" in a bit of the Data Direction Register causes the corresponding peripheral data line to act as an input. During an MPU Read Peripheral Data Operation, the data on peripheral lines programmed to act as inputs appears directly on the corresponding MPU Data Bus lines.

The data in Output Register A will appear on the data lines that are programmed to be outputs. A logical "1" written into the register will cause a "High" on the corresponding data line while a "0" results in a "Low". Data in Output Register A may be read by an MPU "Read Peripheral Data A" operation when the corresponding lines are programmed as outputs.

- **Port B Peripheral Data ($PB_0 \sim PB_7$)**

Each of the Port B peripheral data bus can be programmed to act as an input or output like $PA_0 \sim PA_7$.

$PB_0 \sim PB_7$ are in High-impedance condition because they are three-state outputs just like $PA_0 \sim PB_0$ when the peripheral buses are used as inputs, when programmed as outputs, MPU read of Port B make it possible to read the output register regardless of $PB_0 \sim PB_7$ loads.

- **Interrupt Input (CA_1 and CB_1)**

Peripheral Input lines CA_1 and CB_1 are input only lines that set the interrupt flags of the control registers. The active transition for these signals is also programmed by the two control registers.

- **Peripheral Control (CA_2)**

The peripheral control line CA_2 can be programmed to act as an interrupt input or as a peripheral control output.

The function of this signal is programmed by the Control Register A. When used as an input, this signal is in High-impedance state

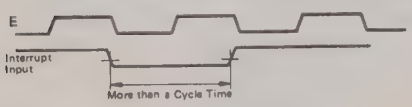
- **Peripheral Control (CB_2)**

Peripheral Control line CB_2 may also be programmed to act as an interrupt input or peripheral control output.

This line is programmed by Control Register B.

When used as an input, this signal is in High-impedance.

(NOTE) 1. Pulse width of interrupt inputs CA_1, CA_2, CB_1 and CB_2 shall be greater than a E cycle time. In the case that "High" time of E signal is not contained in Interrupt pulse, an interrupt flag may not be set.



■ INTERNAL CONTROLS

There are six locations within the PIA accessible to the MPU data bus: two Peripheral Registers, two Data Direction Registers, and two Control Registers. Selection of these locations is controlled by the RS_0 and RS_1 inputs together with bit 2 in the Control Register, as shown in Table 1.

Table 1 Internal Addressing

RS_1	RS_0	Control Register Bit CRA2	Control Register Bit CRB2	Location Selected
0	0	1	x	Peripheral Register A*
0	0	0	x	Data Direction Register A
0	1	x	x	Control Register A
1	0	x	1	Peripheral Register B*
1	0	x	0	Data Direction Register B
1	1	x	x	Control Register B

x = Don't Care
* Peripheral interface register is a generic term containing peripheral data bus and output register.

- **Initialization**

A "Low" reset line has the effect of zeroing all PIA registers. This will set $PA_0 \sim PA_7$, $PB_0 \sim PB_7$, CA_2 and CB_2 as inputs, and all interrupts disabled. The PIA must be configured during the restart program which follows the reset.

Details of possible configurations of the Data Direction and Control Register are as follows.

- **Data Direction Registers (DDRA and DDRB)**

The two Data Direction Registers allow the MPU to control the direction of data through each corresponding peripheral data line. A Data Direction Register bit set at "0" configures the corresponding peripheral data line as an input; a "1" results in an output.

- **Control Registers (CRA and CRB)**

The two Control Registers (CRA and CRB) allow the MPU to control the operation of the four peripheral control lines CA_1, CA_2, CB_1 and CB_2. In addition they allow the MPU to enable the interrupt lines and monitor the status of the interrupt flags. Bits 0 through 5 of the two registers may be written or read by the MPU when the proper chip select and register select signals are applied. Bits 6 and 7 of the two registers are read only and are modified by external interrupts occurring on control lines CA_1, CA_2, CB_1 or CB_2. The format of the control words is shown in Table 2.

Table 2 Control Word Format

	7	6	5	4	3	2	1	0
CRA	IRQA1	IRQA2	CA_2 Control			DDRA Access	CA_1 Control	
CRB	IRQB1	IRQB2	CB_2 Control			DDRB Access	CB_1 Control	

Data Direction Access Control Bit (CRA2 and CRB2)

Bit 2 in each Control register (CRA and CRB) allows selection of either a Peripheral Interface Register or the Data Direction Register when the proper register select signals are applied to RS_0 and RS_1.

Interrupt Flags (CRA6, CRA7, CRB6, and CRB7)

The four interrupt flag bits are set by active transitions of signals on the four Interrupt and Peripheral Control lines when those lines are programmed to be inputs. These bits cannot be set directly from the MPU Data Bus and are reset indirectly by a Read Peripheral Data Operation on the appropriate section.

Control of CA_1 and CB_1 Interrupt Lines (CRA0, CRB0, CRA1, and CRB1)

The two lowest order bits of the control registers are used to control the interrupt input lines CA_1 and CB_1. Bits CRA0 and CRB0 are used to enable the MPU interrupt signals $\overline{IRQA}$ and $\overline{IRQB}$, respectively. Bits CRA1 and CRB1 determine the active transition of the interrupt input signals CA_1 and CB_1 (Table 3).

Control of CA_2 and CB_2 Peripheral Control Lines (CRA3, CRA4, CRA5, CRB3, CRB4, and CRB5)

Bits 3, 4 and 5 of the two control registers are used to control the CA_2 and CB_2 Peripheral Control lines. These bits determine if the control lines will be an interrupt input or an output control signal. If bit CRA5 (CRB5) is "0" CA_2 (CB_2) is an interrupt input line similar to CA1 (CB1) (Table 4). When CRA5 (CRB5) is "1", CA_2 (CB_2) becomes an output signal that may be used to control peripheral data transfers. When in the output mode, CA_2 and CB_2 have slightly different characteristics (Table 5 and 6).

Table 3 Control of Interrupt Inputs CA_1 and CB_1

CRA1 (CRB1)	CRA0 (CRB0)	Interrupt Input CA_1 (CB_1)	Interrupt Flag CRA7 (CRB7)	MPU Interrupt Request $\overline{IRQA}$ ($\overline{IRQB}$)
0	0	↓ Active	Set "1" on ↓ of CA_1 (CB_1)	Disabled – $\overline{IRQ}$ remains "High"
0	1	↓ Active	Set "1" on ↓ of CA_1 (CB_1)	Goes "Low" when the interrupt flag bit CRA7 (CRB7) goes "1"
1	0	↑ Active	Set "1" on ↑ of CA_1 (CB_1)	Disabled – $\overline{IRQ}$ remains "High"
1	1	↑ Active	Set "1" on ↑ of CA_1 (CB_1)	Goes "Low" when the interrupt flag bit CRA7 (CRB7) goes "1"

(Notes)
1. ↑ indicates positive transition ("Low" to "High")
2. ↓ indicates negative transition ("High" to "Low")
3. The Interrupt flag bit CRA7 is cleared by an MPU Read of the A Peripheral Register and CRB7 is cleared by an MPU Read of the B Peripheral Register.
4. If CRA0 (CRB0) is "0" when an interrupt occurs (Interrupt disabled) and is later brought "1", $\overline{IRQA}$ ($\overline{IRQB}$) occurs after CRA0 (CRB0) is written to a "1".

Table 4 Control of CA_2 and CB_2 as Interrupt Inputs – CRA5 (CRB5) is "0"

CRA5 (CRB5)	CRA4 (CRB4)	CRA3 (CRB3)	Interrupt Input CA_2 (CB_2)	Interrupt Flag CRA6 (CRB6)	MPU Interrupt Request $\overline{IRQA}$ ($\overline{IRQB}$)
0	0	0	↓ Active	Set "1" on ↓ of CA_2 (CB_2)	Disabled – $\overline{IRQ}$ remains "High"
0	0	1	↓ Active	Set "1" on ↓ of CA_2 (CB_2)	Goes "Low" when the interrupt flag bit CRA6 (CRB6) goes "1"
0	1	0	↑ Active	Set "1" on ↑ of CA_2 (CB_2)	Disabled – $\overline{IRQ}$ remains "High"
0	1	1	↑ Active	Set "1" on ↑ of CA_2 (CB_2)	Goes "Low" when the interrupt flag bit CRA6 (CRB6) goes "1"

(Notes)
1. ↑ indicates positive transition ("Low" to "High")
2. ↓ indicates negative transition ("High" to "Low")
3. The interrupt flag bit CRA6 is cleared by an MPU Read of the A Peripheral Register and CRB6 is cleared by an MPU Read of the B Peripheral Register.
4. If CRA3 (CRB3) is "0" when an interrupt occurs (Interrupt disabled) and is later brought "1", $\overline{IRQA}$ ($\overline{IRQB}$) occurs after CRA3 (CRB3) is written to a "1".

Table 5 Control of CB_2 as an Output — CRB5 is "1"

CRB5	CRB4	CRB3	CB_2	
			Cleared	Set
1	0	0	"Low" on the positive transition of the first E pulse after MPU Write "B" Data Register operation.	"High" when the interrupt flag bit CRB7 is set by an active transition of the CB_1 signal. (See Figure 16)
1	0	1	"Low" on the positive transition of the first E pulse after an MPU Write "B" Data Register operation.	"High" on the positive edge of the first "E" pulse following an "E" pulse which occurred while the part was deselected. (See Figure 16)
1	1	0	"Low" (The content of CRB3 is output on CB_2)	
1	1	1	"High" (The content of CRB3 is output on CB_2)	

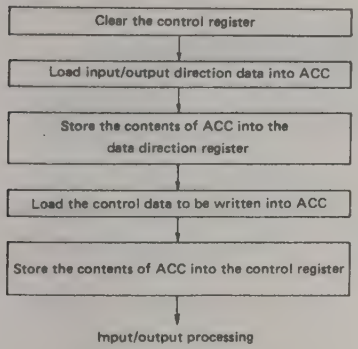
Table 6 Control of CA_2 as an Output — CRA5 is "1"

CRA5	CRA4	CRA3	CA_2	
			Cleared	Set
1	0	0	"Low" on negative transition of E after an MPU Read "A" Data Operation.	"High" when the interrupt flag bit CRA7 is set by an active transition of the CA_1 signal. (See Figure 16)
1	0	1	"Low" on negative transition of E after an MPU Read "A" Data operation.	"High" on the negative edge of the first "E" pulse which occurs during a deselect. (See Figure 16)
1	1	0	"Low" (The content of CRA3 is output on CA_2)	
1	1	1	"High" (The content.of CRA3 is output on CA_2)	

■ PIA OPERATION

• Initialization

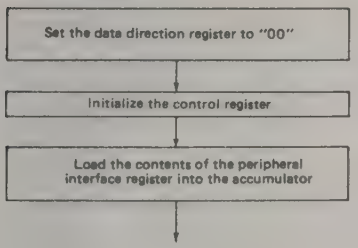
When the external reset input $\overline{RES}$ goes "Low", all internal registers are cleared to "0". Periperal data port (PA_0~PA7, PB_0~PB_7) is defined to be input and control lines (CA_1, CA_2, CB_1 and CB_2) are defined to be the interrupt input lines. PIA is also initialized by software sequence as follows.


Clear the control register
Load input/output direction data into ACC
Store the contents of ACC into the data direction register
Load the control data to be written into ACC
Store the contents of ACC into the control register
Input/output processing

- Program the data direction register access bit of the control register to "0" to allow to access the dada direction register.
- The data of the control line function is set into the accumulator, of which Data Direction Register Access Bit shall be programmed to "1".
- Transfer the control data from the accumulator into the control register.

• Read/Write Operation Not Using Control Lines

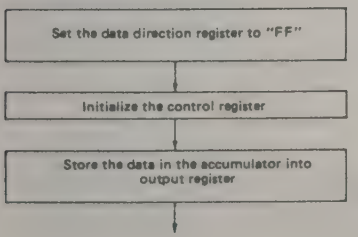
<Read Operation>


Set the data direction register to "00"
Initialize the control register
Load the contents of the peripheral interface register into the accumulator

```
CLR   CRA
CLR   DDRA
LDAA  #$04
STAA  CRA
LDAA  PIRA
```

- Clear the DDRA access bit of the control register to "0".
- Clear all bits of the dada direction register.
- Set DDRA access bit of the control register to "1" to allow to access the peripheral interface register.

<Write Operation>


Set the data direction register to "FF"
Initialize the control register
Store the data in the accumulator into output register

```
CLR   CRA
LDAA  #$FF
STAA  DDRB
LDAA  #$04
STAA  CRB
LDAA  DATA
STAA  PIRB
```

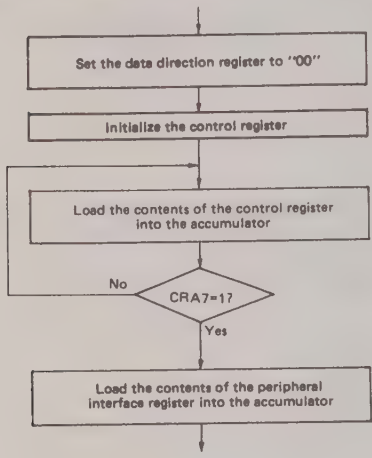
- Set DDRB access bit of the control register to "0".
- Set all bits of the data direction register to "FF".
- Set DDRB access bit of the control register to "1" to allow to access the peripheral interface register.
- Write the data into the peripheral interface register.

• **Read/Write Operating Using Control Lines**

Read/write request from peripherals shall be put into the control lines as an interrupt signal, and then MPU reads or writes after detecting interrupt request.

< Read >

The following case is that Port A is used and that the rising edge of CA_1 indicates the request for read from peripherals.


Set the data direction register to "00"
Initialize the control register
Load the contents of the control register into the accumulator
No
CRA7=1?
Yes
Load the contents of the peripheral interface register into the accumulator

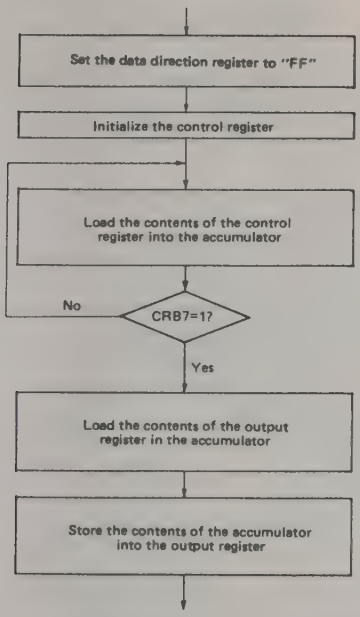
```
CLR   CRA    • Set the DDRA access bit to "0".
CLR   DDRA   • Set all bits of the data direction register to "0".

LDAA  #$06   • Program the rising edge of CA1 to be active. IRQA is masked
STAA  CRA      and DDRA access bit is set to 1.

LOOP  LDAA CRA  }
      BPL LOOP  }  • Check whether the read request comes from peripherals
                     or not.

LDAA  PIRA   • Load the data from the peripheral interface register into the
               accumulator. CRA flag is reset after this read operation.
```

To read the peripheral data, the data is directly transfered to the data buses D_0~D_7 through PA_0~PA_7 or PB_0~PB_7 and they are not latched in the PIA. If necessary, the data should be held in the external latch until MPU completes reading it.

When initializing the control register, interrupt flag bit (CRA7, CRA6, CRB7, CRB6) cannot be written from MPU. If necessary the interrupt flag must be reset by dummy read of Peripheral Register A and B.

<Write>

Write operation using the interrupt signal is as follows. In this case, B port is used and interrupt request is input to CB_1. And the $\overline{IRQ}$ flag is set at the rising edge of CB_1.


Set the data direction register to "FF"
Initialize the control register
Load the contents of the control register into the accumulator
No
CRB7=1?
Yes
Load the contents of the output register in the accumulator
Store the contents of the accumulator into the output register

```
        CLR   CRB      • Set DDRB access bit to "0"
        LDAA  #$FF   }
        STAA  DDRB   }  • Set all bits of DDRB to output "1".

        LDAA  #$06   }
        STAA  CRB    }  • Program the rising edge of CB1 to be active. IRQB is masked
                           and DDRB access bit is set to "1".

LOOP    LDAA  CRB    }
        BPL   LOOP   }  • Check whether the write request comes from peripherals
                           or not.

        LDAA  PIRB      • Reset the CRB7 flag by the dummy read of the
                          peripheral interface register.

        STAB  PIRB      • Store the data of the accumulator B to the peripheral
                          interface register.
```

Interrupt request flag bits (CRA7, CRA6, CRB7 and CRB6) cannot be written and they cannot be also reset by write operation to the peripheral interface register. So dummy read of peripheral interface register is needed to reset the flags.

To accept the next interrupt, it is essential to reset indirectly the interrupt flag by dummy read of peripheral interface register.

Software poling method mentioned above requires MPU to continuously monitor the control register to detect the read/write request from peripherals. So other programs cannot run at the same time. To avoid this problem, hardware interrupt may be used. The MPU is interrupted by $\overline{IRQA}$ or $\overline{IRQB}$ when the read/write request is occurred from peripherals and then MPU analyzes cause of the interrupt request during interrupt processing.

- **Handshake Mode**

The functions of CRA and CRB are similar but not identical in the hand-shake modes. Port A is used for read hand-shake operation and Port B is used for write hand-shake mode.

CA_1 and CB_1 are used for interrupt input requests and CA_2 and CB_2 are control outputs (answer) in hand-shake mode.

Fig. 16, Fig. 17 and Fig. 18 show the timing of hand-shake mode.

< Read Hand-shake Mode>

CRA5="1", CRA4="0" and CRA3="0"

① A peripheral device puts the 8-bit data on the peripheral data lines after the control output CA_2 goes "Low".

② The peripheral requests MPU to read the data by using CA_1 input.

③ CRA7 flag is set and CA_2 becomes "High" (CA_2 automatically becomes "High" by the interrupt CA_1). This indicates the peripheral to maintain the current data and not to transfer the next data.

④ MPU accepts the read request by $\overline{IRQA}$ hardware interrupt or CRA read. Then MPU reads the peripheral register A.

⑤ CA_2 goes "Low" on the following edge of read Enable pulse. This informs that the peripheral can set the next data to port A.

<Write Hand-shake>

CRB5 = "1", CRB4 = "0" and CRB3 = "0"

① A peripheral device requests MPU to write the data by using CB_1 input. CB_2 output remains "High" until MPU write data to the peripheral interface register.

② CRB7 flag is set and MPU accepts the write request.

③ MPU reads the peripheral interface register to reset CRB7 (dummy read).

④ Then MPU write data to the peripheral interface register. The data is output to port B through the output register.

⑤ CB_2 automatically becomes "Low" to tell the peripheral that new data is on port B.

⑥ The peripheral read the data on Port B peripheral data lines and set CB_1 to "Low" to tell MPU that the data on the peripheral data lines has been taken and that next data can be written to the peripheral interface register.

<Pulse mode>

CRA5 = "1", CRA4 = "0" and CRA3 = "1"
CRB5 = "1", CRB4 = "0" and CRB3 = "1"
This mode is shown in Figure 16, Figure 19 and Figure 20

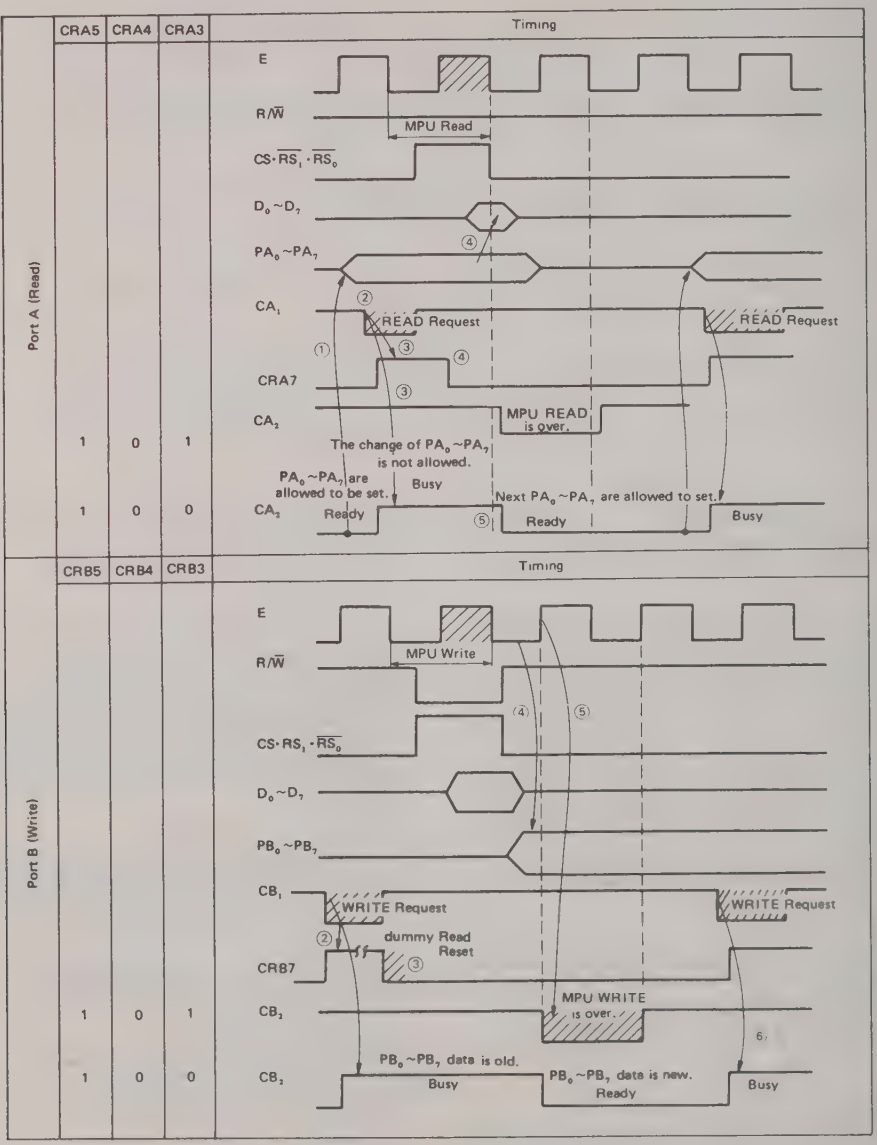


Figure 16 Timing of Hand-shake Mode and Pulse Mode

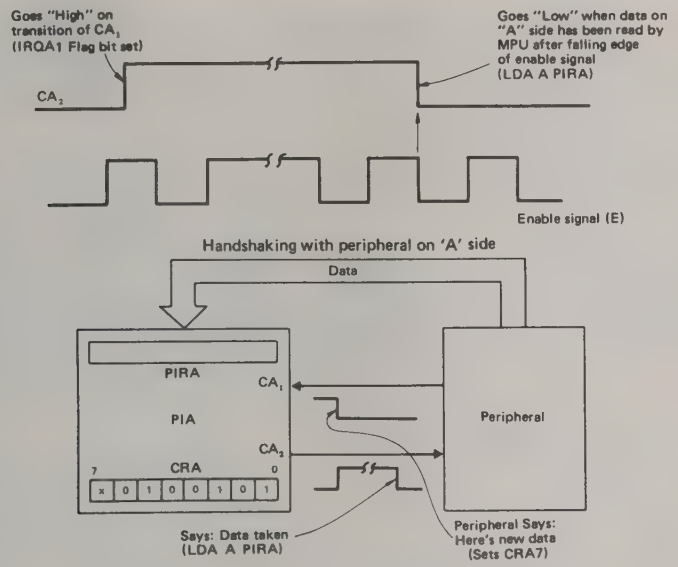

Goes "High" on transition of CA1 (IRQA1 Flag bit set)
Goes "Low" when data on "A" side has been read by MPU after falling edge of enable signal (LDA A PIRA)
CA2
Enable signal (E)
Handshaking with peripheral on 'A' side
Data
PIRA
CA1
PIA
Peripheral
CA2
7
CRA
0
x 0 1 0 0 1 0 1
Says: Data taken (LDA A PIRA)
Peripheral Says: Here's new data (Sets CRA7)

Figure 17 Bits 5, 4, 3 of CRA = 100 (Hand-shake Mode)

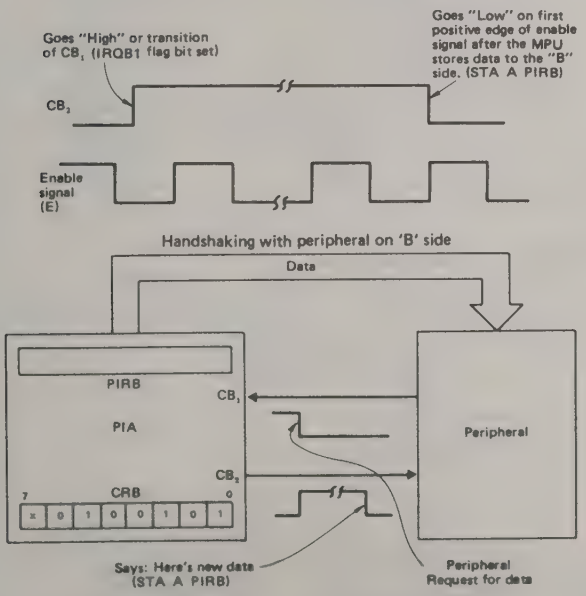

Goes "High" or transition of CB1 (IRQB1 flag bit set)
Goes "Low" on first positive edge of enable signal after the MPU stores data to the "B" side. (STA A PIRB)
CB2
Enable signal (E)
Handshaking with peripheral on 'B' side
Data
PIRB
CB1
PIA
Peripheral
CB2
7
CRB
0
x 0 1 0 0 1 0 1
Says: Here's new data (STA A PIRB)
Peripheral Request for data

Figure 18 Bits 5, 4, 3 of CRB = 100 (Hand-shake Mode)

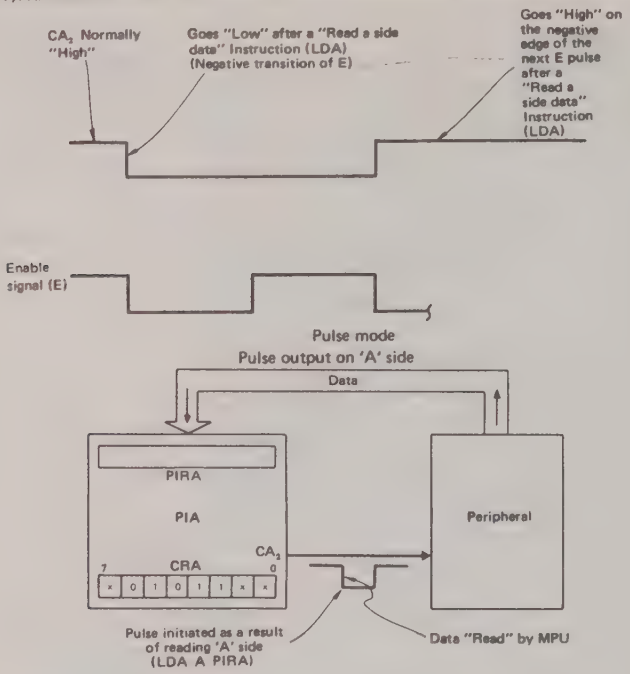

CA2 Normally "High"
Goes "Low" after a "Read a side data" Instruction (LDA) (Negative transition of E)
Goes "High" on the negative edge of the next E pulse after a "Read a side data" Instruction (LDA)
Enable signal (E)
Pulse mode
Pulse output on 'A' side
Data
PIRA
PIA
CRA
CA2
7
0
x 0 1 0 1 1 x x
Peripheral
Pulse initiated as a result of reading 'A' side (LDA A PIRA)
Data "Read" by MPU

Figure 19 Bits 5, 4, 3 of CRA = 101 (Pulse Mode)

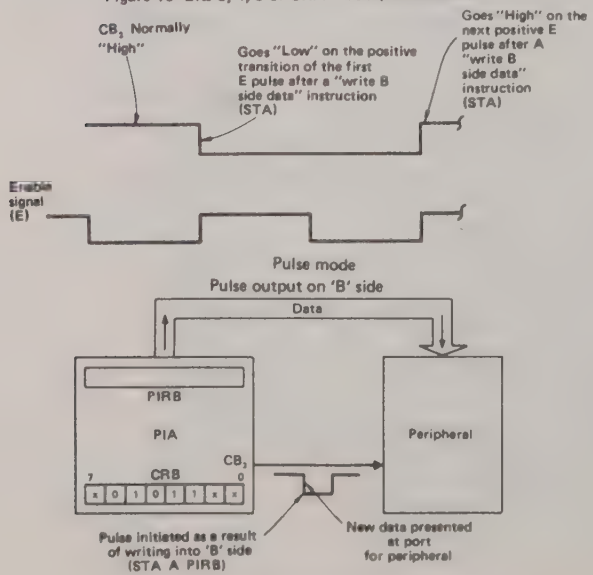

CB2 Normally "High"
Goes "Low" on the positive transition of the first E pulse after a "write B side data" instruction (STA)
Goes "High" on the next positive E pulse after A "write B side data" instruction (STA)
Enable signal (E)
Pulse mode
Pulse output on 'B' side
Data
PIRB
PIA
CRB
CB2
7
0
x 0 1 0 1 1 x x
Peripheral
Pulse initiated as a result of writing into 'B' side (STA A PIRB)
New data presented at port for peripheral

Figure 20 Bits 5, 4, 3 of CRB=101 (Pulse Mode)

■ SUMMARY OF CONTROL REGISTERS CRA AND CRB

Control registers CRA and CRB have total control of CA_1, CA_2, CB_1, and CB_2 lines. The status of eight bits of the control registers may be read into the MPU. However, the MPU can only write into Bit 0 through Bit 5 (6 bits), since Bit 6 and Bit 7 are set only by CA_1, CA_2, CB_1, or CB_2.

● Addressing PIAs

Before addressing PIAs, the data direction (DDR) must first be loaded with the bit pattern that defines how each line is to function, i.e., as an input or an output. A logic "1" in the data direction register defines the corresponding line as an output while a logic "0" defines the corresponding line as an input. Since the DDR and the peripheral interface resister have the same address, the control register bit 2 determines which register is being addressed. If Bit 2 in the control register is a logic "0", then the DDR is addressed. If Bit 2 in the control register is a logic "1", the peripheral interface register is addressed. Therefore, it is essential that the DDR be loaded first before setting Bit 2 of the control register.

<Example>

Given a PIA with an address of 4004, 4005, 4006, and 4007. 4004 is the address of the A side peripheral interface register. 4005 is the address of the A side control register. 4006 is the address of the B side peripheral interface register. 4007 is the address of the B side control register. On the A side, Bits 0, 1, 2, and 3 will be defined as inputs, while Bits 4, 5, 6, and 7 will be used as outputs. On the B side, all lines will be used as outputs.

```
PIA1AD = 4004      (DDRA, PIRA)
PIA1AC = 4005      (CRA)
PIA1BD = 4006      (DDRB, PIRB)
PIA1BC = 4007      (CRB)

1.  LDA A #%11110000   (4 outputs, 4 inputs)
2.  STA A    PIA1AD    (Loads A DDR)
3.  LDA A #%11111111   (All outputs)
4.  STA A    PIA1BD    (Loads B DDR)
5.  LDA A #%00000100   (Sets Bit 2)
6.  STA A    PIA1AC    (Bit 2 set in A control register)
7.  STA A    PIA1BC    (Bit 2 set in B control register)
```

Statement 2 addresses the DDR, since the control register (Bit 2) has not been loaded. Statements 6 and 7 load the control registers with Bit 2 set, so addressing PIA1AD or PIA1BD accesses the peripheral interface register.

● PIA Programming Via The Index Register

The program shown in the previous section can be accomplished using the Index Register.

```
1.  LDX   #$F004
2.  STX   PIA1AD     $F0→PIA1AD ;$04→PIA1AC
3.  LDX   #$FF04
4.  STX   PIA1BD     $FF→PIA1BD ;$04→PIA1BC
```

Using the index register in this example has saved six bytes of program memory as compared to the program shown in the previous section.

● Active Low Outputs

When all the outputs of given PIA port are to be active "Low" (True ≦ 0.4 volts), the following procedure should be used.

a) Set Bit 2 in the control register.
b) Store all 1s ($FF) in the peripheral interface register.
c) Clear Bit 2 in the control register.
d) Store all 1s ($FF) in the data direction register.
e) Store control word (Bit 2 = 1) in control register.

<Example>

The B side of PIA1 is set up to have all active low outputs. CB_1 and CB_2 are set up to allow interrupts in the HANDSHAKE MODE and CB_1 will respond to positive edges ("Low"-to-"High" transitions). Assume reset conditions. Addresses are set up and equated to the same labels as previous example.

```
1.  LDA A #4
2.  STA A PIA1BC     Set Bit 2 in PIA1BC (control register)
3.  LDA B #$FF
4.  STA B PIA1BD     All 1s in peripheral interface register
5.  CLR PIA1BC       Clear Bit 2
6.  STA B PIA1BD     All 1s in data direction register
7.  LDA A #$27
8.  STA A PIA1BC     00100111→→ control register
```

The above procedure is required in order to avoid outputs going "Low", to the active "Low" TRUE STATE, when all ls are stored to the data direction register as would be the case if the normal configuration procedure were followed.

● Interchanging RS_0 And RS_1

Some system applications may require movement of 16 bits of data to or from the "outside world" via two PIA ports (A side + B side). When this is the case it is an advantage to interconnect RS_1 and RS_0 as follows.

RS_0 to A1 (Address Line A1)
RS_1 to A0 (Address Line A0)

This will place the peripheral interface registers and control registers side by side in the memory map as follows.

Table	Example Address	
PIA1AD	$4004	(DDRA, PIRA)
PIA1BD	$4005	(DDRB, PIRB)
PIA1AC	$4006	(CRA)
PIA1BC	$4007	(CRB)

The index register or stackpointer may be used to move the 16-bit data in two 8-bit bytes with one instruction. As an example:

LDX PIA1AD PIA1AD →→ IX_H: PIA1BD →→ IX_L

● PIA – After Reset

When the $\overline{RES}$ (Reset Line) has been held "Low" for a minimum of one microsecond, all registers in the PIA will be cleared.

Because of the reset conditions, the PIA has been defined as

follows.

1. All I/O lines to the "outside world" have been defined as inputs.
2. CA_1, CA_2, CB_1, and CB_2 have been defined as interrupt input lines that are negative edge sensitive.
3. All the interrupts on the control lines are masked. Setting of interrupt flag bits will not cause $\overline{IRQA}$ or $\overline{IRQB}$ to go "Low".

■ SUMMARY OF CA_1-CB_1 PROGRAMMING

Bits 1 and 0 of the respective control registers are used to program the interrupt input control lines CA_1 and CB_1.

b1	b0	
0	0	b1 = Edge (0 = –, 1 = +)
0	1	b0 = Mask (0 = Mask, 1 = Allow)
1	0	
1	1	

Note that this is the same logic as Bits 4 and 3 for CA_2-CB_2 when CA_2-CB_2 are programmed as inputs.

■ SUMMARY OF CA_2-CB_2 PROGRAMMING

Bits 5, 4, and 3 of the control registers are used to program the operation of CA_2-CB_2.

	b5	b4	b3		
CA_2–CB_2 Input Mode →	0	0(–)	0	(Mask)	CA_2–CB_2 Input Mode
	0	0(–)	1	(Allow)	b4 = Edge (0 = –, 1 = +)
	0	1(+)	0	(Mask)	b3 = Mask (0 = Mask,
	0	1(+)	1	(Allow)	1 = Allow)
CA_2–CB_2 Output Mode →	1	0		0 – Handshake Mode	
	1	0		1 – Pulse Mode	
	1	1	0	b3 Following Mode	
	1	1	1		

I/O As Follow:
Control Lines:
CA_1 – Positive Edge, Allow Interrupt
CA_2 – Pulse Mode
CB_1 – Negative Edge, Mask Interrupt
CB_2 – Hand Shake Mode

Assume Reset Condition
PIA1AD
PIA1AC
PIA1BD
PIA1BC

PIA Configuration Solution

LDA A #$BC	10111100
STA A PIA1AD	I/O to DDRA
LDA A #$FF	1111 1111
STA A PIA1BD	I/O to DDRB
LDA A #$2F	0010 1111
STA A PIA1AC	To "A" Control
LDA A #$24	0010 0100
STA A PIA1BC	To "B" Control

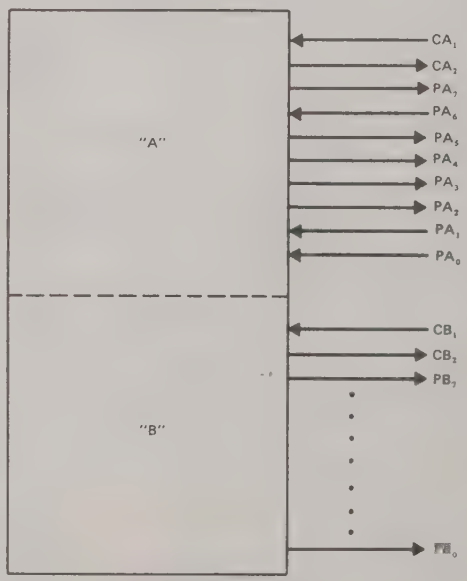
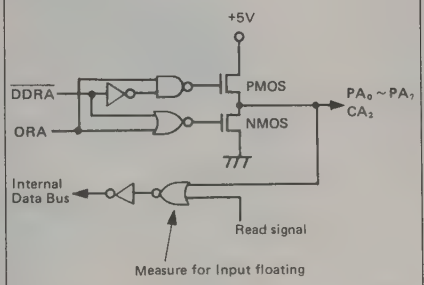
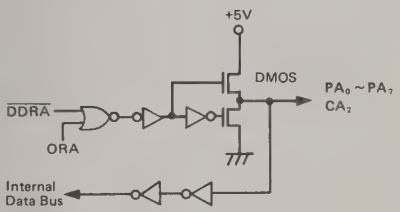
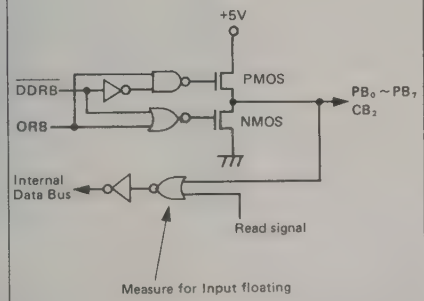
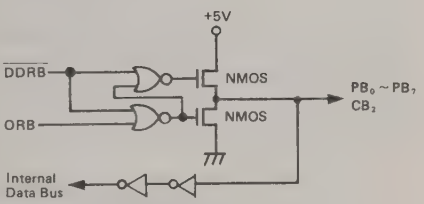


Figure 21 PIA Configuration Problem

■ NOTE FOR USE

Compatibility with NMOS PIA (HD6821)

Table 7 Comparison CMOS PIA (HD6321) with NMOS PIA (HD6821)

Item	CMOS PIA (HD6321)	NMOS PIA (HD6821)
Port A Output Buffer	Three-state output 	Pull-up output 
Port B Output Buffer	Three-state output 	Three-state output 

There is no difference between CMOS PIA and NMOS PIA in pin arrangement.

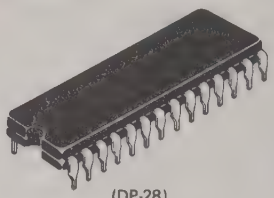
HD6840, HD68A40, HD68B40

PTM (Programmable Timer Module)

The HD6840 is a programmable subsystem component of the HMCS6800 family designed to provide variable system time intervals.

The HD6840 has three 16-bit binary counters, three corresponding control registers and a status register. These counters are under software control and may be used to cause system interrupts and/or generate output signals. The HD6840 may be utilized for such tasks as frequency measurements, event counting, interval measuring and similar tasks. The device may be used for square wave generation, gated delay signals, single pulses of controlled duration, and pulse width modulation as well as system interrupts.

HD6840P, HD68A40P, HD68B40P

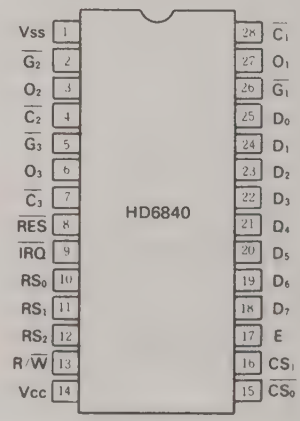


(DP-28)

■ FEATURES

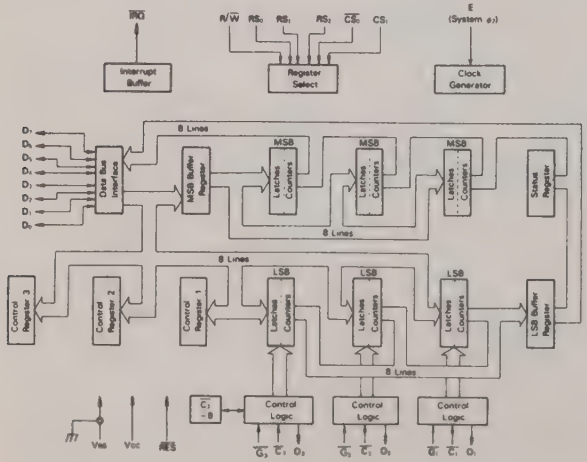
- Operates from a Single 5 Volts Power Supply
- Fully TTL Compatible
- Single System Clock Required (E)
- Selectable Prescaler on Timer 3 Capable of 4 MHz for the HD6840, 6 MHz for the HD68A40 and 8 MHz for the HD68B40
- Programmable Interrupts ($\overline{IRQ}$) Output to MPU
- Readable Down Counter Indicates Counts to Go until Time-Out
- Selectable Gating for Frequency or Pulse-Width Comparison
- $\overline{RES}$ Input
- Three Asynchronous External Clock and Gate/Trigger Inputs Internally Synchronized
- Three Maskable Outputs
- Compatible with MC6840, MC68A40 and MC68B40

■ PIN ARRANGEMENT



(Top View)

■ BLOCK DIAGRAM



HITACHI

■ ABSOLUTE MAXIMUM RATINGS

Item	Symbol	Value	Unit
Supply Voltage	V_{CC}*	-0.3~+7.0	V
Input Voltage	V_{in}*	-0.3~+7.0	V
Operating Temperature	T_{opr}	- 20~+ 75	°C
Storage Temperature	T_{stg}	- 55~+150	°C

* With respect to V_{SS} (SYSTEM GND)

[NOTE] Permanent LSI damage may occur if maximum ratings are exceeded. Normal operation should be under recommended operating conditions. If these conditions are exceeded, it could affect reliability of LSI.

■ RECOMMENDED OPERATING CONDITIONS

Item	Symbol	min	typ	max	Unit
Supply Voltage	V_{CC}*	4.75	5.0	5.25	V
Input Voltage	V_{IL}*	-0.3	–	0.8	V
	V_{IH}*	2.2	–	V_{CC}	V
Operating Temperature	T_{opr}	- 20	25	75	°C

* With respect to V_{SS} (SYSTEM GND)

■ ELECTRICAL CHARACTERISTICS

• DC CHARACTERISTICS (V_{CC} = 5V ± 5%, V_{SS} = 0V, Ta = -20 ~ +75°C, unless otherwise noted.)

Item	Symbol	Test Condition		min	typ*	max	Unit
Input "High" Voltage	V_{IH}			2.2	–	V_{CC}	V
Input "Low" Voltage	V_{IL}			-0.3	–	0.8	V
Input Leakage Current	I_{in}	V_{in} = 0 ~ 5.25V (Except D_0 ~ D_7)		-2.5	–	2.5	μA
Three-State Input Current (off-state)	I_{TSI}	V_{in} = 0.4 ~ 2.4V, V_{CC} = 5.25V (D_0 ~ D_7)		- 10	–	10	μA
Output "High" Voltage	V_{OH}	I_{LOAD} = -205 μA (D_0 ~ D_7)		2.4	–	–	V
		I_{LOAD} = -200 μA (Other Outputs)					
Output "Low" Voltage	V_{OL}	I_{LOAD} = 1.6 mA (D_0 ~ D_7)		–	–	0.4	V
		I_{LOAD} = 3.2 mA (O_1 ~ O_3, $\overline{IRQ}$)					
Output Leakage Current (off-state)	I_{LOH}	V_{OH} = 2.4V ($\overline{IRQ}$)		–	–	10	μA
Power Dissipation	P_D			–	330	550	mW
Input Capacitance	C_{in}	V_{in} = 0V, Ta = 25°C, f = 1 MHz	D_0 ~ D_7	–	–	12.5	pF
			Other Input	–	–	7.5	
Output Capacitance	C_{out}	V_{in} = 0V, Ta = 25°C, f = 1 MHz	$\overline{IRQ}$	–	–	5.0	pF
			O_1, O_2, O_3	–	–	10	

* Ta = 25°C, V_{CC} = 5.0V

- **AC CHARACTERISTICS (V_{CC} = 5V ± 5%, V_{SS} = 0V, Ta = -20 ~ +75°C, unless otherwise noted.)**

1. MPU READ TIMING

Item	Symbol	Test Condition	HD6840			HD68A40			HD68B40			Unit
			min	typ	max	min	typ	max	min	typ	max	
Enable Cycle Time	t_{cycE}	Fig. 1	1.0	–	10	0.666	–	10	0.5	–	10	μs
Enable "High" Pulse Width	PW_{EH}		0.45	–	4.5	0.280	–	4.5	0.22	–	4.5	μs
Enable "Low" Pulse Width	PW_{EL}		0.43	–	–	0.280	–	–	0.21	–	–	μs
Enable Rise and Fall Time	t_{Er}, t_{Ef}		–	–	25	–	–	25	–	–	25	ns
Address Set Up Time	t_{AS}		140	–	–	140	–	–	70	–	–	ns
Data Delay Time	t_{DDR}		–	–	320	–	–	220	–	–	180	ns
Data Hold Time	t_H		10	–	–	10	–	–	10	–	–	ns
Address Hold Time	t_{AH}		10	–	–	10	–	–	10	–	–	ns
Data Access Time	t_{ACC}		–	–	480	–	–	360	–	–	250	ns

2. MPU WRITE TIMING

Item	Symbol	Test Condition	HD6840			HD68A40			HD68B40			Unit
			min	typ	max	min	typ	max	min	typ	max	
Enable Cycle Time	t_{cycE}	Fig. 2	1.0	–	10	0.666	–	10	0.5	–	10	μs
Enable "High" Pulse Width	PW_{EH}		0.45	–	4.5	0.280	–	4.5	0.22	–	4.5	μs
Enable "Low" Pulse Width	PW_{EL}		0.43	–	–	0.280	–	–	0.21	–	–	μs
Enable Rise and Fall Time	t_{Er}, t_{Ef}		–	–	25	–	–	25	–	–	25	ns
Address Set Up Time	t_{AS}		140	–	–	140	–	–	70	–	–	ns
Data Set Up Time	t_{DSW}		195	–	–	80	–	–	60	–	–	ns
Data Hold Time	t_H		10	–	–	10	–	–	10	–	–	ns
Address Hold Time	t_{AH}		10	–	–	10	–	–	10	–	–	ns

3. TIMING OF PTM SIGNAL

Item			Symbol	Test Condition		HD6840		HD68A40		HD68B40		Unit
						min	max	min	max	min	max	
Input Rise and Fall Times	$\overline{C}$, G, $\overline{RES}$		t_r, t_f	Fig. 3, Fig. 4		–	1.0*	–	0.666*	–	0.5*	μs
Input "Low" Pulse Width	$\overline{C}$, G, $\overline{RES}$		PW_L	Fig. 3 (Asynchronous Mode)		$t_{cycE}+t_{SU}+t_{HD}$	–	$t_{cycE}+t_{SU}+t_{HD}$	–	$t_{cycE}+t_{SU}+t_{HD}$	–	ns
Input "High" Pulse Width	$\overline{C}$, G		PW_H	Fig. 4 (Asynchronous Mode)		$t_{cycE}+t_{SU}+t_{HD}$	–	$t_{cycE}+t_{SU}+t_{HD}$	–	$t_{cycE}+t_{SU}+t_{HD}$	–	ns
Input Setup Time	$\overline{C}$, G, $\overline{RES}$		t_{SU}	Fig. 5 (Synchronous Mode)		200	–	120	–	75	–	ns
	$\overline{C_3}$ (÷8 Prescaler Mode)					200	–	170	–	170	–	
Input Hold Time	$\overline{C}$, G, $\overline{RES}$		t_{HD}	Fig. 5 (Synchronous Mode)		50	–	50	–	50	–	ns
	$\overline{C_3}$ (÷8 Prescaler Mode)					50	–	50	–	50	–	
Input Pulse Width	$\overline{C_3}$ (÷8 Prescaler Mode)		PW_L, PW_H	(Asynchronous Mode)		125	–	84	–	62.5	–	ns
Output Delay Time	$O_1 \sim O_3$	TTL	t_{co}	Fig. 6	V_{OH}=2.4V, Load B	–	700	–	460	–	340	ns
		MOS	t_{cm}		V_{OH}=2.4V, Load D	–	450	–	450	–	340	ns
		CMOS	t_{cmos}		V_{OH}=0.7×V_{CC} Load D	–	2.0	–	1.35	–	1.0	μs
Interrupt Release Time			t_{IR}	Fig. 7		–	1.2	–	0.9	–	0.7	μs

* t_r, $t_f \leqq t_{cycE}$

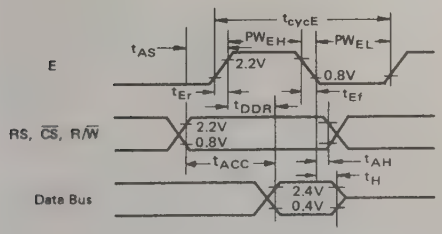


Figure 1 Bus Read Timing
(Read Information from PTM)

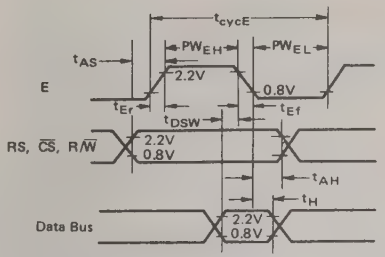


Figure 2 Bus Write Timing
(Write Information into PTM)

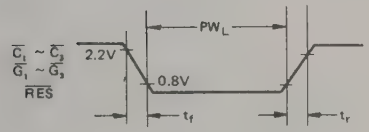


Figure 3 Input Pulse Width "Low"

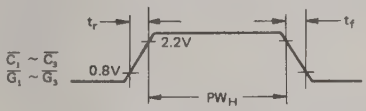


Figure 4 Input Pulse Width "High"

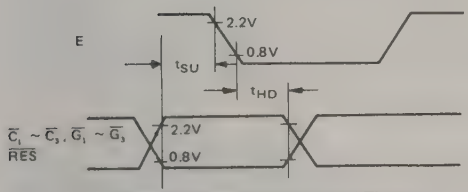


Figure 5 Input Setup and Hold Times

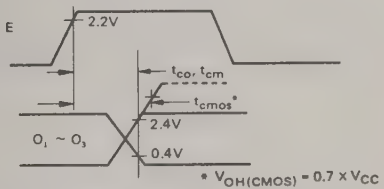


Figure 6 Output Delay

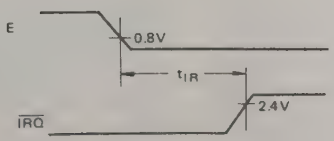


Figure 7 $\overline{IRQ}$ Release Time

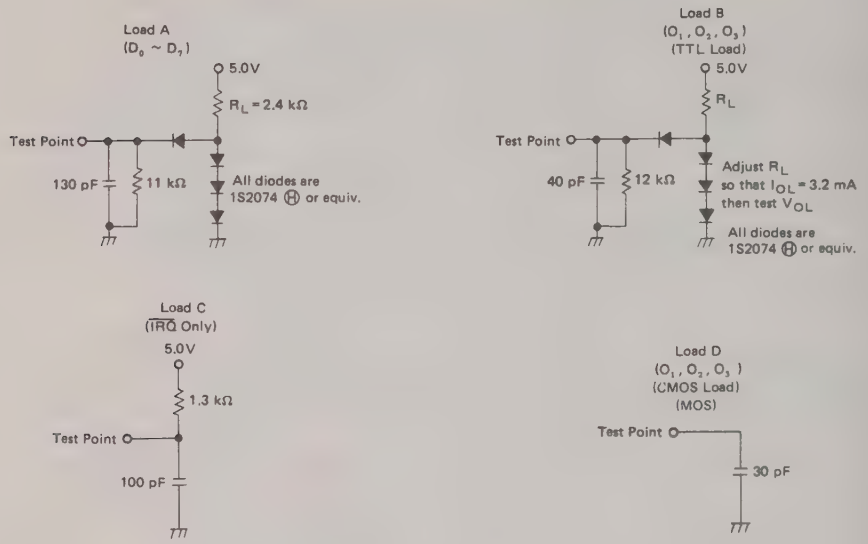


Figure 8 Test Loads

■ GENERAL DESCRIPTION

The HD6840 is part of the HMCS6800 microprocessor family and is fully bus compatible with HD6800 systems. The three timers in the HD6840 operate independently and in several distinct modes to fit a wide variety of measurement and synthesis applications.

The HD6840 is an integrated set of three distinct counter/timers. It consists of three 16-bit data latches, three 16-bit counters (clocked independently), and the comparison and enable circuitry necessary to implement various measurement and synthesis functions. In addition, it contains interrupt drivers to alert the processor that a particular function has been completed.

In a typical application, a timer will be loaded by first storing two bytes of data into an associated Counter Latch. This data is then transferred into the counter via a Counter initialization cycle. If the counter is enabled, the counter decrements on each subsequent clock period which may be an external clock, or Enable (E) until one of several predetermined conditions causes it to halt or recycle. The timers are thus programmable, cyclic in nature, controllable by external inputs or the MPU program, and accessible by the MPU at any time.

■ PTM INTERFACE SIGNALS FOR MPU

The Programmable Timer Module (PTM) interfaces to the HMCS6800 Bus with an eight-bit bidirectional data bus, two Chip Select lines, a Read/Write line, an Enable (System ϕ_2) line, an Interrupt Request line, an external Reset line, and three Register Select lines. These signals, in conjunction with the HD6800 VMA output, permit the MPU to control the PTM. VMA should be utilized in conjunction with an MPU address line into a Chip Select of the PTM, when the HD6800, HD6802 are used.

- **Bidirectional Data ($D_0 \sim D_7$)**

The bidirectional data lines ($D_0 \sim D_7$) allow the transfer of data between the MPU and PTM. The data bus output drivers are three-state devices which remain in the high-impedance (off) state except when the MPU performs a PTM read operation (Read/Write and Enable lines "High" and PTM Chip Selects activated).

- **Chip Select ($\overline{CS_0}$, CS_1)**

These two signals are used to activate the Data Bus interface and allow transfer of data from the PTM. With $\overline{CS_0}$ = "Low" and CS_1 = "High", the device is selected and data transfer will occur.

- **Read/Write (R/$\overline{W}$)**

This signal is generated by the MPU to control the direction of data transfer on the Data Bus. With the PTM selected, a "Low" state on the PTM R/$\overline{W}$ line enables the input buffers and

data is transferred from the MPU to the PTM on the trailing edge of the Enable (System ϕ_2) signal. Alternately, (under the same conditions) R/$\overline{W}$ = "High" and Enable "High" allows data in the PTM to be read by the MPU.

- **Enable (E)**

This signal synchronizes data transfer between the MPU and the PTM. It also performs an equivalent synchronization function on the external clock, reset, and gate inputs of the PTM.

- **Interrupt Request ($\overline{IRQ}$)**

The active "Low" Interrupt Request signal is normally tied directly (or through priority interrupt circuitry) to the $\overline{IRQ}$ input of the MPU. This is an "open drain" output (no load device on the chip) which permits other similar interrupt request lines to be tied together in a wire-OR configuration.

The $\overline{IRQ}$ line is activated if, and only if, the Composite Interrupt Flag (Bit 7 of the Internal Status Register) is asserted. The conditions under which the $\overline{IRQ}$ line is activated are discussed in conjunction with the Status Register.

- **Reset ($\overline{RES}$)**

A "Low" level at this input is clocked into the PTM by the Enable (System ϕ_2) input. Two Enable pulses are required to synchronize and process the signal. The PTM then recognizes the active "Low" or inactive "High" on the third Enable pulse. If the $\overline{RES}$ signal is asynchronous, an additional Enable period is required if setup times are not met. The $\overline{RES}$ input must be stable "High"/"Low" for the minimum time stated in the AC Characteristics.

Recognition of a "Low" level at this input by the PTM causes the following action to occur:

a. All counter latches are preset to their maximal count values.
b. All Control Register bits are cleared with the exception of CR10 (internal reset bit) which is set.
c. All counters are preset to the contents of the latches.
d. All counter outputs are reset and all counter clocks are disabled.
e. All Status Register bits (interrupt flags) are cleared.

- **Register Select Lines (RS_0, RS_1, RS_2)**

These inputs are used in conjunction with the R/$\overline{W}$ line to select the internal registers, counters and latches as shown in Table 1.

It has been previously stated that the PTM is accessed via MPU Load and Store operations in much the same manner as a memory device. The instructions available with the HMCS6800 family of MPUs which perform operations directly on memory should not be used when the PTM is accessed. These instructions actually fetch a byte from memory, perform an operation, then restore it to the same address location. Since the PTM used the R/$\overline{W}$ line as an additional register select input, the modified data may not be restored to the same register if these instructions are used.

■ PTM ASYNCHRONOUS INPUT/OUTPUT SIGNALS

Each of the three timers within the PTM has external clock and gate inputs as well as a counter output line. The inputs are high impedance, TTL compatible lines and outputs are capable of driving two standard TTL loads.

- **Clock Inputs ($\overline{C_1}$, $\overline{C_2}$, $\overline{C_3}$)**

Input pins $\overline{C_1}$, $\overline{C_2}$, and $\overline{C_3}$ will accept asynchronous TTL voltage level signals to decrement Timers 1, 2, and 3, respectively. The "High" and "Low" levels of the external clocks must each be stable for at least one system clock period plus the sum

Table 1 Register Selection

Register Select Inputs *			Operations	
RS_2	RS_1	RS_0	R/$\overline{W}$ = "Low"	R/$\overline{W}$ = "High"
L	L	L	CR20 = "0" Write Control Register #3 CR20 = "1" Write Control Register #1	No Operation
L	L	H	Write Control Register #2	Read Status Register
L	H	L	Write MSB Buffer Register	Read Timer #1 Counter
L	H	H	Write Timer #1 Latches	Read LSB Buffer Register
H	L	L	Write MSB Buffer Register	Read Timer #2 Counter
H	L	H	Write Timer #2 Latches	Read LSB Buffer Register
H	H	L	Write MSB Buffer Register	Read Timer #3 Counter
H	H	H	Write Timer #3 Latches	Read LSB Buffer Register

* L; "Low" level, H; "High" level

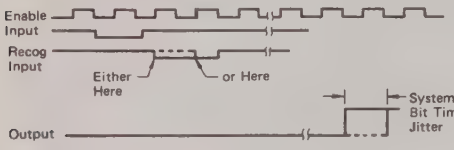
of the setup and hold times for the inputs. The asynchronous clock rate can vary from dc to the limit imposed by Enable (System ϕ_2) Setup, and Hold time.

The external clock inputs are clocked in by Enable (System ϕ_2) pulses. Three Enable periods are used to synchronize and process the external clock. The fourth Enable pulse decrements the internal counter. This does not affect the input frequency, it merely creates a delay between a clock input transition and internal recognition of that transition by the PTM. All references to $\overline{C}$ inputs in this document relate to internal recognition of the input transition. Note that a clock "High" or "Low" level which does not meet setup and hold time specifications may require an additional Enable pulse for recognition. When observing recurring events, a lack of synchronization will result in "jitter" being observed on the output of the PTM when using asynchronous clocks and gate input signals. There are two types of jitter. "System jitter" is the result of the input signals being out of synchronization with the Enable (System ϕ_2), permitting signals with marginal setup and hold time to be recognized by either the bit time nearest the input transition or the subsequent bit time.

"Input jitter" can be as great as the time between input signal negative going transitions plus the system jitter, if the first transition is recognized during one system cycle, and not recognized the next cycle, or vice versa.

External clock input $\overline{C_3}$ represents a special case when Timer

#3 is programmed to utilize its optional ÷8 prescaler mode. The maximum input frequency and allowable duty cycles for this case are specified under the AC Characteristics. The output of the ÷8 prescaler is treated in the same manner as the previously discussed clock inputs. That is, it is clocked into the counter by Enable pulses, is recognized on the fourth Enable pulse (provided setup and hold time requirements are met), and must produce an output pulse at least as wide as the sum of an Enable period, setup, and hold times.



- **$\overline{\text{Gate}}$ Inputs ($\overline{G_1}$, $\overline{G_2}$, $\overline{G_3}$)**

Input pins $\overline{G_1}$, $\overline{G_2}$, and $\overline{G_3}$ accept asynchronous TTL-compatible signals which are used as triggers or clock gating functions to Timers 1, 2, and 3, respectively. The gating inputs are clocked into the PTM by the Enable (System ϕ_2) signal in the same manner as the previously discussed clock inputs. That is, a $\overline{\text{Gate}}$ transition is recognized by the PTM on the fourth Enable pulse (provided setup and hold time requirements are met), and the "High" or "Low" levels of the $\overline{\text{Gate}}$ input must be stable for at least one system clock period plus the sum of setup and hold times. All references to $\overline{\text{G}}$ transition in this document relate to internal recognition of the input transition.

The $\overline{\text{Gate}}$ inputs of all timers directly affected the internal 16-bit counter. The operation of $\overline{G_3}$ is therefore independent of the ÷8 prescaler selection.

- **Timer Outputs (O_1, O_2, O_3)**

Timer outputs O_1, O_2, and O_3 are capable of driving up to two TTL loads and produce a defined output waveform for either Continuous or Single-Shot Timer modes. Output waveform definition is accomplished by selecting either Single 16-bit or Dual 8-bit operating modes. The single 16-bit mode will produce a square-wave output in the continuous timer mode and will produce a single pulse in the Single-Shot Timer mode. The Dual 8-bit mode will produce a variable duty cycle pulse in both the continuous and single shot Timer modes. "1" bit of each Control Register (CRX7) is used to enable the corresponding output. If this bit is cleared, the output will remain "Low" (V_{OL}) regardless of the operating mode.

If it is cleared while the output is high the output will go low during the first enable cycle following a write to the Control Register.

The Continuous and Single-Shot Timer Modes are the only ones for which output response is defined in this data sheet. Signals appear at the outputs (unless CRX7 = "0") during Frequency and Pulse Width comparison modes, but the actual waveform is not predictable in typical applications.

■ CONTROL REGISTER

Each timer in the HD6840 has a corresponding write-only Control Register. Control Register #2 has a unique address space (RS0="High", RS1="Low", RS2="Low") and therefore may be written into at any time. The remaining Control Registers (#1 and#3) share the Address Space selected by a "Low" level on all Register Select inputs.

- **CR20**

The least-significant bit of Control Register #2 (CR20) is used as an additional addressing bit for Control Registers #1 and #3. Thus, with all Register selects and R/$\overline{\text{W}}$ inputs at "Low" level. Control Register #1 will be written into if CR20 is a logic "1". Under the same conditions, control Register #3 can also be written into after a $\overline{\text{RES}}$ "Low" condition has occurred, since all control register bits (except CR10) are cleared. Therefore, one may write in the sequence CR3, CR2, CR1.

- **CR10**

The least-significant bit of Control Register #1 is used as an internal Reset bit. When this bit is a logic "0", all timers are allowed to operate in the modes prescribed by the remaining bits of the control registers. Writing a "1" into CR10 causes all counters to be preset with the contents of the corresponding counter latches, all counter clocks to be disabled, and the timer outputs and interrupt flags (Status Register) to be reset. Counter Latches and Control Registers are undisturbed by an Internal Reset and may be written into regardless of the state of CR10.

- **CR30**

The least-significant bit of Control Register #3 is used as a selector for a ÷8 prescaler which is available with Timer #3 only. The prescaler, if selected, is effectively placed between the clock input circuitry and the input to Counter #3. It can therefore be used with either the internal clock (Enable) or an external clock source.

- **CRX1 ~ CRX7 (X=1~3)**

The functions depicted in the foregoing discussions are tabulated in Table 2 for ease of reference.

Control Register Bits CR10, CR20, and CR30 are unique in that each selects a different function. The remaining bits (1 through 7) of each Control Register select common functions, with a particular Control Register affecting only its corresponding timer.

- **CRX1**

Bit 1 of Control Register #1 (CR11) selects whether an internal or external clock source is to be used with Timer #1. Similarly, CR21 selects the clock source for Timer #2, and CR31 performs this function for Timer #3. The function of each bit of Control Register "X" can therefore be defined as shown in the remaining section of Table 2.

- **CRX2**

Control Register Bit 2 selects whether the binary information contained in the Counter Latches (and subsequently loaded into the counter) is to be treated as a single 16-bit word or two 8-bit bytes. In the single 16-bit Counter Mode (CRX2=0) the counter will decrement to zero after N + 1 enabled ($\overline{\text{G}}$="Low") clock periods, where N is defined as the 16-bit number in the Counter Latches. With CRX2 = 1, a similar Time Out will occur after (L + 1)·(M + 1) enabled clock periods, where L and M, respectively, refer to the LSB and MSB bytes in the Counter Latches.

- **CRX3 ~ CRX7**

Control Register Bits 3, 4, and 5 are explained in detail in the Timer Operating Mode section. Bit 6 is an interrupt mask bit which will be explained more fully in conjunction with the Status Register, and bit 7 is used to enable the corresponding Timer Output. A summary of the control register programming modes is shown in Table 3.

■ STATUS REGISTER/INTERRUPT FLAGS

The HD6840 has an internal Read-Only Status Register which contains four Interrupt Flags. (The remaining four bits of the register are not used, and default to "0"s when being read.) Bits 0, 1, and 2 are assigned to Timers 1, 2, and 3, respectively, as individual flag bits, while Bit 7 is a Composite Interrupt Flag. This flag bit will be asserted if any of the individual flag bits is

Table 2 Control Register Bits

CONTROL REGISTER #1		CONTROL REGISTER #2		CONTROL REGISTER #3	
CR10	Internal Reset Bit	CR20	Control Register Address Bit	CR30	Timer #3 Clock Control
"0" All timers allowed to operate "1" All timers held in preset state		"0" CR #3 may be written "1" CR #1 may be written		"0" T3 Clock is not prescaled "1" T3 Clock is prescaled by ÷ 8	
CRX1*		Timer #X Clock Source			
"0"		TX uses external clock source on $\overline{CX}$ input			
"1"		TX uses Enable clock			
CRX2		Timer #X Counting Mode Control			
"0"		TX configured for normal (16-bit) counting mode			
"1"		TX configured for dual 8-bit counting mode			
CRX3 CRX4 CRX5		Timer #X Counter Mode and Interrupt Control (See Table 3)			
CRX6		Timer #X Interrupt Enable			
"0"		Interrupt Flag masked on $\overline{IRQ}$			
"1"		Interrupt Flag enabled to $\overline{IRQ}$			
CRX7		Timer #X Counter Output Enable			
"0"		TX Output masked on output OX			
"1"		TX Output enabled on output OX			

* Control Register for Timer 1, 2, or 3, Bit 1.

set while Bit 6 of the corresponding Control Register is at a logic "1". The conditions for asserting the Composite Interrupt Flag bit can therefore be expressed as:

$$INT = I_1 \cdot CR16 + I_2 \cdot CR26 + I_3 \cdot CR36$$

where INT = Composite Interrupt Flag (Bit 7)
I_1 = Timer #1 Interrupt Flag (Bit 0)
I_2 = Timer #2 Interrupt Flag (Bit 1)
I_3 = Timer #3 Interrupt Flag (Bit 2)

STATUS REGISTER

7	6	5	4	3	2	1	0
INT					I_3	I_2	I_1

An interrupt flag is cleared by a Timer Reset condition, i.e., External $\overline{RES}$ = "Low" or Internal Reset Bit (CR10) = "1". It will also be cleared by a Read Timer Counter Command provided that the Status Register has previously been read while the interrupt flag was set. This condition on the Read Status Register – Read Timer Counter (RS–RT) sequence is designed to prevent missing interrupts which might occur after the status register is read, but prior to reading the Timer Counter.

An Individual Interrupt Flag is also cleared by a Write Timer Latches (W) command or a Counter Initialization (CI) sequence, provided that W or CI affects the Timer corresponding to the individual Interrupt Flag.

■ COUNTER LATCH INITIALIZATION

Each of the three independent timers consists of a 16-bit addressable counter and 16 bits of addressable latches. The counters are preset to the binary numbers stored in the latches. Counter initialization results in the transfer of the latch contents to the counter. See notes in Table 5 regarding the binary number N, L, or M placed into the Latches and their relationship to the output waveforms and counter Time-Outs.

Since the PTM data bus is 8-bits wide and the counters are 16-bits wide, a temporary register (MSB Buffer Register) is provided. This "write only" register is for the Most Significant Byte of the desired latch data. Three addresses are provided for the MSB Buffer Register (as indicated in Table 1), but they all lead to the same Buffer. Data from the MSB Buffer will automatically be transferred into the Most Significant Byte of Timer #X when a Write Timer #X Latches Command is performed. So it can be seen that the HD6840 has been designed to allow transfer of two bytes of data into the counter latches provided that the MSB is transferred first.

In many applications, the source of the data will be as HMCS6800 MPU. It should be noted that the 16-bit store operations of the HMCS6800 microprocessors (STS and STX etc.) transfer data in the order required by the PTM. A Store Index Register Instruction, for example, results in the MSB of the X register being transferred to the selected address, then the LSB of the X register being written into the next higher location. Thus, either the index register or stack pointer may be transfered directly into a selected counter latch with a single instruction.

A logic "Low" at the $\overline{RES}$ input also initializes the counter latches. In this case, all latches will assume a maximum count of $(65{,}536)_{10}$. It is important to note that an Internal Reset (Bit 0 of Control Register 1 Set) has no effect on the counter latches.

■ COUNTER INITIALIZATION

Counter Initialization is defined as the transfer of data from the latches to the counter with subsequent clearing of the Individual Interrupt Flag associated with the counter. Counter Initialization always occurs when a reset condition ($\overline{RES}$ = "Low" or CR10 = "1") is recognized. It can also occur – depending on Timer Mode – with a Write Timer Latches command or recognition of a negative transition of the Gate input.

Counter recycling or re-initialization occurs when a negative transition of the clock input is recognized after the counter has reached an all-zero state. In this case, data is transferred from the Latches to the Counter.

■ TIMER OPERATING MODES

The HD6840 has been designed to operate effectively in a wide variety of applications. This is accomplished by using three bits of each control register (CRX3, CRX4, and CRX5) to

defined different operating modes of the Timers. These modes are divided into Wave Synthesis and Wave Measurement modes, and outlined in Table 4.

Table 4 Operating Modes

Control Register			Timer Operating Mode	
CRX3	CRX4	CRX5		
0	•	0	Continuous	Wave Synthesis
0	•	1	Single-Shot	
1	0	•	Frequency Comparison	Wave Measurement
1	1	•	Pulse Width Comparison	

* Defines Additional Timer Functions.

One of the WAVE SYNTHESIS modes is the Continuous Operating mode, which is useful for cyclic wave generation. Either symmetrical or variable duty-cycle waves can be generated in this mode. The other wave synthesis mode, the Single-Shot mode, is similar in use to the Continuous operating mode, however, a single pulse is generated, with a programmable preset width.

The WAVE MEASUREMENT modes include the Frequency Comparison and Pulse Width Comparison modes which are used to measure cyclic and singular pulse widths, respectively.

In addition to the four timer modes in Table 4, the remaining control register bit is used to modify counter initialization and enabling or interrupt conditions.

■ WAVE SYNTHESIS MODES

● Continuous Operating Mode (Table 5)

The continuous mode will synthesize a continuous wave with

Table 3 Control Register Programming

7	6	5	4	3	2	1	0	Register 1	Register 2	Register 3
X	X	X	X	X	X	X	↕	"0" All Timers Operate	Reg #3 May Be Written	T3 Clk ÷ 1
								"1" All Timers Preset	Reg #1 May Be Written	T3 Clk ÷ 8

7	6	5	4	3	2	1	0	
X	X	X	X	X	X	↕	X	"0" External Clock ($\overline{CX}$ Input) "1" Internal Clock (Enable)

7	6	5	4	3	2	1	0	
X	X	X	X	X	↕	X	X	"0" Normal (16-Bit) Count Mode "1" Dual 8-Bit Count Mode

7	6	5	4	3	2	1	0	
X	X	0	0	0	X	X	X	Continuous Operating Mode: $\overline{Gate}$ ↓ or Write to Latches or Reset Causes Counter Initialization

7	6	5	4	3	2	1	0	
X	X	0	0	1	X	X	X	Frequency Comparison Mode: Interrupt if $\overline{Gate}$ ↑↓↑ is < Counter Time Out

7	6	5	4	3	2	1	0	
X	X	0	1	0	X	X	X	Continuous Operating Mode: $\overline{Gate}$ ↓ or Reset Causes Counter Initialization

7	6	5	4	3	2	1	0	
X	X	0	1	1	X	X	X	Pulse Width Comparison Mode: Interrupt if $\overline{Gate}$ ↑↓↑ is < Counter Time Out

7	6	5	4	3	2	1	0	
1	X	1	0	0	X	X	X	Single Shot Mode: $\overline{Gate}$ ↓ or Write to Latches or Reset Causes Counter Initialization

7	6	5	4	3	2	1	0	
X	X	1	0	1	X	X	X	Frequency Comparison Mode: Interrupt If $\overline{Gate}$ ↑↓↑ is > Counter Time Out

7	6	5	4	3	2	1	0	
1	X	1	1	0	X	X	X	Single Shot Mode: $\overline{Gate}$ ↓ or Reset Causes Counter Initialization

7	6	5	4	3	2	1	0	
X	X	1	1	1	X	X	X	Pulse Width Comparison Mode: Interrupt If $\overline{Gate}$ ↑↓↑ is > Counter Time Out

7	6	5	4	3	2	1	0	
X	↕	X	X	X	X	X	X	"0" Interrupt Flag Masked ($\overline{IRQ}$) "1" Interrupt Flag Enabled ($\overline{IRQ}$)

7	6	5	4	3	2	1	0	
↕	X	X	X	X	X	X	X	"0" Timer Output Masked "1" Timer Output Enable

(NOTE) Reset is Hardware or Software Reset ($\overline{RES}$ = "Low" or CR10 = "1").

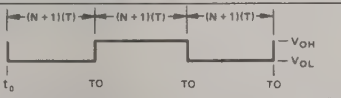
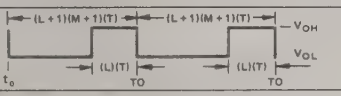
a period proportional to the preset number in the particular timer latches.

Any of the timers in the PTM may be programmed to operate in a continuous mode by writing "0"s into bits 3 and 5 of the corresponding control register. Assuming that the timer output is enabled (CRX7 = "1"), either a square wave or a variable duty cycle waveform will be generated at the Timer Output, OX. The type of output is selected via Control Register Bit 2.

Either a Timer Reset (CR10 = "1" or External $\overline{RES}$ = "Low") condition or internal recognition of a negative transition of the $\overline{Gate}$ input results in Counter Initialization. A Write Timer Latches command can be selected as a Counter Initialization signal by clearing CRX4.

The counter is enabled by an absence of a Timer Reset condition and a "Low" level at the $\overline{Gate}$ input. In the 16-bit mode, the counter will decrement on the first clock cycle during or after the counter initialization cycle. It continues to decrement on each clock signal so long as $\overline{G}$ remains "Low" and no reset condition exists. A Counter Time Out (the first clock after all

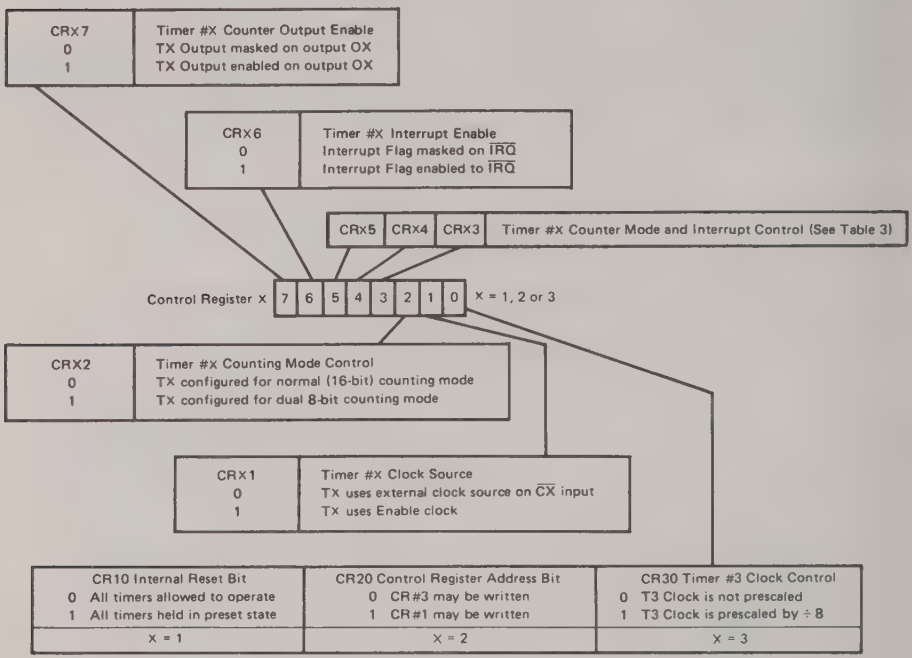
Table 5 Continuous Operating Modes

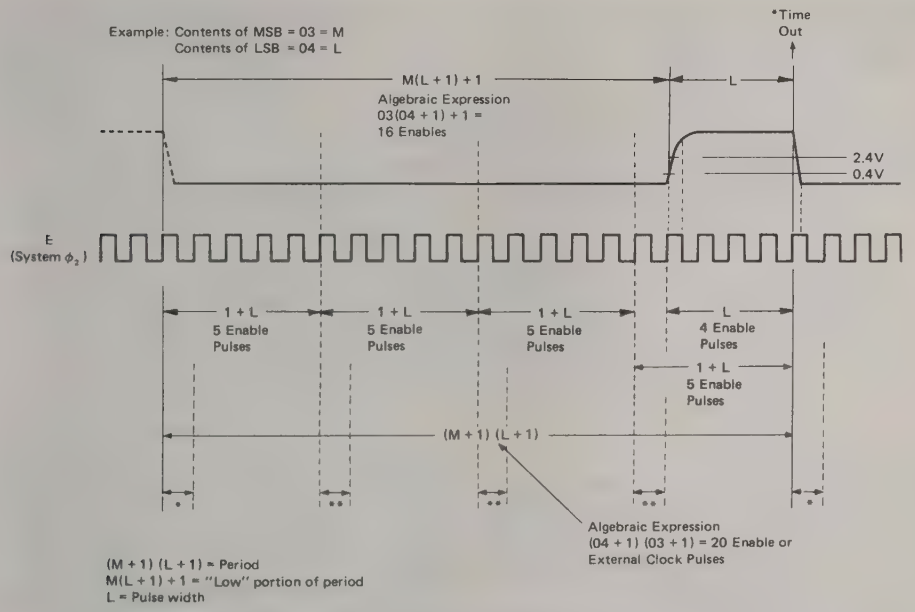
CONTINUOUS MODE (CRX3 = "0", CRX5 = "0")			
Control Register		Initialization/Output Waveforms	
CRX2	CRX4	Counter Initialization	*Timer Output (OX) (CRX7 = "1")
0	0	$\overline{G}\downarrow$+W+R	
0	1	$\overline{G}\downarrow$+R	
1	0	$\overline{G}\downarrow$+W+R	
1	1	$\overline{G}\downarrow$+R	

$\overline{G}\downarrow$ = Negative transition of $\overline{Gate}$ input.
W = Write Timer Latches Command.
R = Timer Reset (CR10 = "1" or External $\overline{RES}$ = "Low")
N = 16-Bit Number in Counter Latch.
L = 8-Bit Number in LSB Counter Latch.
M = 8-Bit Number in MSB Counter Latch.
T = Clock Input Negative Transitions to Counter.
t_0 = Counter Initialization Cycle.
TO = Counter Time Out (All Zero Condition).

* All time intervals shown above assume the $\overline{Gate}$ ($\overline{G}$) and $\overline{Clock}$ ($\overline{C}$) signals are synchronized to Enable (System ϕ_2) with the specified setup and hold time requirements.

Control Register Bits





* Preset LSB and MSB to Respective Latches on the negative transition of the E.
** Preset LSB to LSB Latches and Decrement MSB by one on the negative transition of the E.

Figure 9 Timer Output Waveform Example
(Continuous Dual 8-Bit Mode using Internal Enable)

counter bits = "0") results in the Individual Interrupt Flag being set and re-initialization of the counter.

In the dual 8-bit mode (CRX2 = "1") [Refer to the example in Fig. 9] the MSB decrements once for every full countdown of the LSB + 1. When the LSB = "0", the MSB is unchanged; on the next clock pulse the LSB is reset to the count in the LSB Latches and the MSB is decremented by 1 (one). The output, if enabled, remains "Low" during and after initialization and will remain "Low" until the counter MSB is all "0"s. The output will go "High" at the beginning of the next clock pulse. The output remains "High" until both the LSB and MSB of the counter are all "0"s. At the beginning of the next clock pulse the defined Time Out (TO) will occur and the output will go "Low". In the Dual 8-bit mode the period of the output of the example in Fig. 9 would span 20 clock pulses as opposed to the 1546 clock pulses using the Normal 16-bit mode.

A special time-out condition exists for the dual 8-bit mode (CRX2 = "1") if L = "0". In this case, the counter will revert to a mode similar to the single 16-bit mode, except Time Out occurs after M+1 clock pulses. The output, if enabled, goes "Low" during the Counter Initialization cycle and reverses state at each Time Out. The counter remains cyclical (is re-initialized at each Time Out) and the Individual Interrupt Flag is set when Time Out occurs. If M = L = "0", the internal counters do not change, but the output toggles at a rate of 1/2 the clock frequency.

The discussion of the Continuous Mode has assumed that the application requires an output signal. It should be noted that the Timer operates in the same manner with the output disabled (CRX7 = "0"). A Read Timer Counter command is valid regardless of the state of CRX7.

- **Single-Shot Timer Mode**

This mode is identical to the Continuous Mode with three exceptions. The first of these is obvious from the name – the output returns to a "Low" level after the initial Time Out and remains "Low" until another Counter Initialization cycle occurs. The waveforms available are shown in Table 6.

As indicated in Table 6, the internal counting mechanism remains cyclical in the Single-Shot Mode. Each Time Out of the counter results in the setting of an Individual Interrupt Flag and re-initialization of the counter.

The second major difference between the Single-Shot and Continuous modes is that the internal counter enable is not dependent on the $\overline{\text{Gate}}$ input level remaining in the "Low" state for the Single-Shot mode.

Another special condition is introduced in the Single-Shot mode. If L = M = "0"(Dual 8-bit) or N = "0" (Single 16-bit), the output goes "Low" on the first clock received during or after Counter Initialization. The output remains "Low" until the Operating Mode is changed or nonzero data is written into the Counter Latches. Time Outs continue to occur at the end of each clock period.

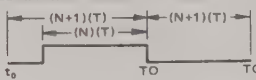
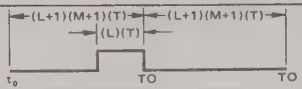
The three differences between Single-Shot and Continuous Timer Modes can be summarized as attributes of the Single-Shot

mode:

1. Output is enabled for only one pulse until it is reinitialized.
2. Counter Enable is independent of $\overline{Gate}$.
3. L = M = "0" or N = "0" disables output.

Aside from these differences, the two modes are identical.

Table 6 Single-Shot Operating Modes

Single-Shot Mode (CRX3 = "0", CRX7 = "1", CRX5 = "1")			
Control Register		Initialization/Output Waveforms	
CRX2	CRX4	Counter Initialization	Timer Output (OX)
0	0	$\overline{G}\downarrow$+W+R	
0	1	$\overline{G}\downarrow$+R	
1	0	$\overline{G}\downarrow$+W+R	
1	1	$\overline{G}\downarrow$+R	

Symbols are as defined in Table 5.

■ Wave Measurement Modes

The Wave Measurement Modes are the Frequency (period) Measurement and Pulse Width Comparison Modes, and are provided for those applications which require more flexibility of interrupt generation and Counter Initialization. Individual Interrupt Flags are set in these modes as a function of both Counter Time Out and transitions of the $\overline{Gate}$ input. Counter Initialization is also affected by Interrupt Flag status.

A timer's output is normally not used in a Wave Measurement mode, but it is defined. If the output is enabled, it will operate as follows. During the period between reinitialization of the timer and the first Time Out, the output will be a logical zero. If the first Time Out is completed (regardless of its method of generation), the output will go "High". If further TO's occur, the output will change state at each completion of a Time-Out.

The counter does operate in either Single 16-bit or Dual 8-bit modes as programmed by CRX2. Other features of the Wave Measurement Modes are outlined in Table 7.

Table 7 Wave Measurement Modes

CRX3 = "1"			
CRX4	CRX5	Application	Condition for Setting Individual Interrupt Flag
0	0	Frequency Comparison	Interrupt Generated if $\overline{Gate}$ Input Period (1/F) is less than Counter Time Out (TO)
0	1	Frequency Comparison	Interrupt Generated if $\overline{Gate}$ Input Period (1/F) is greater than Counter Time Out (TO)
1	0	Pulse Width Comparison	Interrupt Generated if $\overline{Gate}$ Input "Down Time" is less than Counter Time Out (TO)
1	1	Pulse Width Comparison	Interrupt Generated if $\overline{Gate}$ Input "Down Time" is greater than Counter Time Out (TO)

• Frequency Comparison or Period Measurement Mode (CRX3 = "1", CRX4 = "0")

The Frequency Comparison Mode with CRX5 = "1" is straightforward. If Time Out occurs prior to the first negative transition of the $\overline{Gate}$ input after a Counter Initialization cycle, an Individual Interrupt Flag is set. The counter is disabled, and a Counter Initialization cycle cannot begin until the interrupt flag is cleared and a negative transition on $\overline{G}$ is detected.

If CRX5 = "0", as shown in Table 7 and Table 8, an interrupt is generated if $\overline{Gate}$ input returns "Low" prior to a Time Out. If Counter Time-Out occurs first, the counter is recycled and continues to decrement. A bit is set within the timer on the initial Time Out which precludes further individual interrupt generation until a new Counter Initialization cycle has been completed. When this internal bit is set, a negative transition of the $\overline{Gate}$ input starts a new Counter Initialization cycle. (The condition of $\overline{G}\downarrow \cdot \overline{1} \cdot TO$ is satisfied, since a Time Out has occurred and no individual Interrupt has been generated.)

Any of the timers within the PTM may be programmed to compare the period of a pulse (giving the frequency after calculations) at the Gate input with the time period requested for Counter Time-Out. A negative transition of the $\overline{Gate}$ input enables the counter and starts a Counter Initialization cycle – provided that other conditions as noted in Table 8 are satisfied. The counter decrements on each clock signal recognized during or after Counter Initialization until an Interrupt is generated, a Write Timer Latches command is issued, or a Timer Reset condition occurs. It can be seen from Table 8 that an interrupt condition will be generated if CRX5 = "0" and the period of the pulse (single pulse or measured separately repetitive pulses) at the $\overline{Gate}$ input is less than the Counter Time Out period. If CRX5 = "1", an interrupt is generated if the reverse is true.

Assume now with CRX5 = "1" that a Counter Initialization has occurred and that the $\overline{\text{Gate}}$ input has returned "Low" prior to Counter Time Out. Since there is no Individual Interrupt Flag generated, this automatically starts a new Counter Initialization Cycle. The process will continue with frequency comparison being performed on each $\overline{\text{Gate}}$ input cycle until the mode is changed, or a cycle is determined to be above the predetermined limit.

- **Pulse Width Comparison Mode (CRX3 = "1", CRX4 = "1")**

This mode is similar to the Frequency Comparison Mode except for a positive, rather than negative, transition of the $\overline{\text{Gate}}$ input terminates the count. With CRX5 = "0", an Individual Interrupt Flag will be generated if the "Low" level pulse applied to the $\overline{\text{Gate}}$ input is less than the time period required for Counter Time Out. With CRX5 = "1", the interrupt is generated when the reverse condition is true.

As can be seen in Table 9, a positive transition of the $\overline{\text{Gate}}$ input disables the counter. With CRX5 = "0", it is therefore possible to directly obtain the width of any pulse causing an interrupt. Similar data for other Time Interval Modes and conditions can be obtained, if two sections of the PTM are dedicated to the purpose.

Table 8 Frequency Comparison Mode

CRX3 = "1", CRX4 = "0"				
Control Reg Bit 5 (CRX5)	Counter Initialization	Counter Enable Flip-Flop Set (CE)	Counter Enable Flip-Flop Reset (CE)	Interrupt Flag Set (I)
0	$\overline{G}\downarrow \cdot \overline{I} \cdot (\overline{CE}+TO)+R$	$\overline{G}\downarrow \cdot \overline{W} \cdot \overline{R} \cdot \overline{I}$	W+R+I	$\overline{G}\downarrow$ Before TO
1	$\overline{G}\downarrow \cdot \overline{I}+R$	$\overline{G}\downarrow \cdot \overline{W} \cdot \overline{R} \cdot \overline{I}$	W+R+I	TO Before $\overline{G}\downarrow$

I represents the interrupt for a given timer.

Table 9 Pulse Width Comparison Mode

CRX3 = "1", CRX4 = "1"				
Control Reg Bit 5 (CRX5)	Counter Initialization	Counter Enable Flip-Flop Set (CE)	Counter Enable Flip-Flop Reset (CE)	Interrupt Flag Set (I)
0	$\overline{G}\downarrow \cdot \overline{I}+R$	$\overline{G}\downarrow \cdot \overline{W} \cdot \overline{R} \cdot \overline{I}$	W+R+I+G	$\overline{G}\uparrow$ Before TO
1	$\overline{G}\downarrow \cdot \overline{I}+R$	$\overline{G}\downarrow \cdot \overline{W} \cdot \overline{R} \cdot \overline{I}$	W+R+I+G	TO Before $\overline{G}\uparrow$

G = Level sensitive recognition of $\overline{\text{Gate}}$ input.

HD6340,HD63A40,HD63B40
CMOS PTM (Programmable Timer Module)

—PRELIMINARY—

The HD6340 is a programmable subsystem component of the HMCS6800 family designed to provide variable system time intervals.

The HD6340 has three 16-bit binary counters, three corresponding control registers and a status register. These counters are under software control and may be used to cause system interrupts and/or generate output signals. The HD6340 may be utilized for such tasks as frequency measurements, event counting, interval measuring and similar tasks. The device may be used for square wave generation, gated delay signals, single pulses of controlled duration, and pulse width modulation as well as system interrupts.

Exceeding Low Power dissipation is realized due to adopting CMOS process.

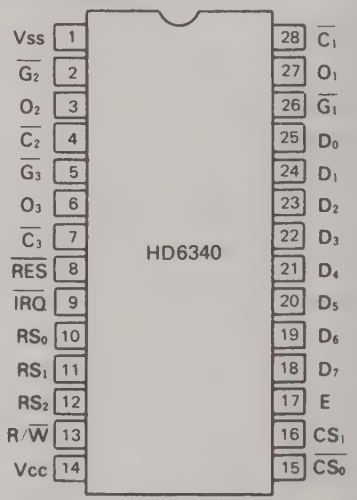
HD6340P,HD63A40P,HD63B40P

(DP-28)

■ FEATURES

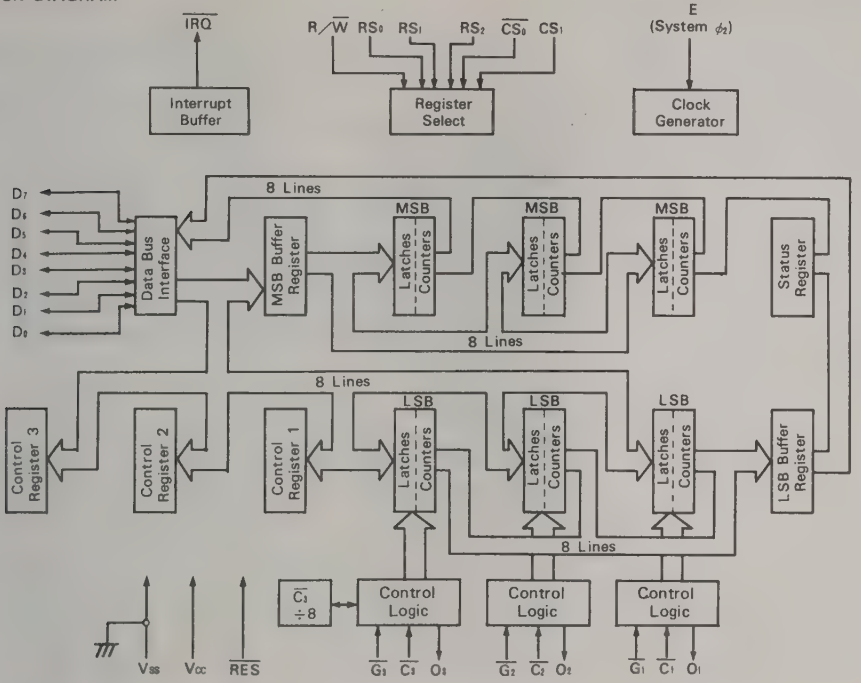
- Operates from a Single 5 Volts Power Supply
- Fully TTL Compatible
- Single System Clock Required (E)
- Selectable Prescaler on Timer 3 Capable of 4 MHz for the HD6340, 6 MHz for the HD63A40 and 8 MHz for the HD63B40
- Programmable Interrupts ($\overline{IRQ}$) Output to MPU
- Readable Down Counter Indicates Counts to Go until Time-Out
- Selectable Gating for Frequency or Pulse-Width Comparison
- $\overline{RES}$ Input
- Three Asynchronous External Clock and Gate/Trigger Inputs Internally Synchronized
- Three Maskable Outputs
- Low-Power, High-Speed, High-Density CMOS
- Compatible with NMOS PTM (HD6840)

■ PIN ARRANGEMENT



(Top View)

■ BLOCK DIAGRAM



■ ABSOLUTE MAXIMUM RATINGS

Item	Symbol	Value	Unit
Supply Voltage	V_{CC}*	-0.3~+7.0	V
Input Voltage	V_{in}*	-0.3~+7.0	V
Maximum Output Current	$\|I_{IO}\|$**	10	mA
Operating Temperature	T_{opr}	- 20~+75	°C
Storage Temperature	T_{stg}	- 55~+150	°C

* With respect to V_{SS} (SYSTEM GND)

** Maximum output current is the maximum currents which can flow out from one output terminal or I/O common terminal. ($D_0 \sim D_7$, $O_1 \sim O_3$, $\overline{IRQ}$)

[NOTE] Permanent LSI damage may occur if maximum ratings are exceeded. Normal operation should be under recommended operating conditions. If these conditions are exceeded, it could affect reliability of LSI.

■ RECOMMENDED OPERATING CONDITIONS

Item	Symbol	min	typ	max	Unit
Supply Voltage	V_{CC}*	4.5	5.0	5.5	V
Input Voltage	V_{IL}*	0	–	0.8	V
	V_{IH}*	2.2	–	V_{CC}	V
Operating Temperature	T_{opr}	- 20	25	75	°C

* With respect to V_{SS} (SYSTEM GND)

■ ELECTRICAL CHARACTERISTICS

● DC CHARACTERISTICS (V_{CC} = 5V ±10%, V_{SS} = 0V, Ta = -20 ~ +75°C, unless otherwise noted.)

Item	Symbol	Test Condition		min	typ*	max	Unit
Input "High" Voltage	V_{IH}			2.2	–	V_{CC}	V
Input "Low" Voltage	V_{IL}			-0.3	–	0.8	V
Input Leakage Current	I_{in}	V_{in} = 0 ~ V_{CC} (Except D_0 ~ D_7)		-2.5	–	2.5	μA
Three-State Input Current (off-state)	I_{TSI}	V_{in} = 0.4 ~ V_{CC}, V_{CC} = 5.5V (D_0 ~ D_7)		-10	–	10	μA
Output "High" Voltage	V_{OH}	I_{LOAD} = -400 μA (D_0 ~ D_7)		4.1	–	–	V
		$I_{LOAD} \leq$ -10 μA (D_0 ~ D_7)		V_{CC}-0.1	–	–	
		I_{LOAD} = -400 μA (Other Outputs)		4.1	–	–	
		$I_{LOAD} \leq$ -10 μA (Other Outputs)		V_{CC}-0.1	–	–	
Output "Low" Voltage	V_{OL}	I_{LOAD} = 1.6 mA (D_0 ~ D_7)		–	–	0.4	V
		I_{LOAD} = 3.2 mA (O_1 ~ O_3, $\overline{IRQ}$)					
Output Leakage Current (off-state)	I_{LOH}	V_{OH} = V_{CC} ($\overline{IRQ}$)		–	–	10	μA
Supply Current	I_{CC}	• Chip is not selected. • All counter latches are preset. • O_1 ~ O_3 outputs are masked. • Input level (Except E) V_{IH} min = V_{CC}-0.8V V_{IL} max = 0.8V	E = 1.0 MHz	–	–	1.0	mA
			E = 1.5 MHz	–	–	1.5	
			E = 2.0 MHz	–	–	2.0	
		• Chip is not selected. • Counters are operating. • O_1 ~ O_3 operating with load. • Input level (Except E) V_{IH} min = V_{CC}-0.8V V_{IL} max = 0.8V	E = 1.0 MHz	–	–	3.0	
			E = 1.5 MHz	–	–	4.0	
			E = 2.0 MHz	–	–	6.0	
		• Data bus in R/$\overline{W}$ operation. • Counters are operating. • O_1 ~ O_3 operating with load.	E = 1.0 MHz	–	–	5.0	
			E = 1.5 MHz	–	–	8.0	
			E = 2.0 MHz	–	–	10.0	
Input Capacitance	C_{in}	V_{in} = 0V, Ta = 25°C, f = 1 MHz	D_0 ~ D_7	–	–	12.5	pF
			Other Input	–	–	7.5	
Output Capacitance	C_{out}	V_{in} = 0V, Ta = 25°C, f = 1 MHz	$\overline{IRQ}$	–	–	5.0	pF
			O_1, O_2, O_3	–	–	10	

* Ta = 25°C, V_{CC} = 5.0V

• AC CHARACTERISTICS (V_{CC} = 5V ±10%, V_{SS} = 0V, Ta = -20 ~ +75°C, unless otherwise noted.)

1. MPU READ TIMING

Item	Symbol	Test Condition	HD6340			HD63A40			HD63B40			Unit
			min	typ	max	min	typ	max	min	typ	max	
Enable Cycle Time	t_{cycE}	Fig. 1	1000	—	10000	666	—	10000	500	—	10000	ns
Enable "High" Pulse Width	PW_{EH}		450	—	9500	280	—	9500	220	—	9500	ns
Enable "Low" Pulse Width	PW_{EL}		430	—	9500	280	—	9500	210	—	9500	ns
Enable Rise and Fall Time	t_{Er}, t_{Ef}		—	—	25	—	—	25	—	—	20	ns
Address Set Up Time	t_{AS}		80	—	—	60	—	—	40	—	—	ns
Data Delay Time	t_{DDR}		—	—	290	—	—	180	—	—	150	ns
Data Hold Time	t_{HR}		20	—	100	20	—	100	20	—	100	ns
Address Hold Time	t_{AH}		10	—	—	10	—	—	10	—	—	ns
Data Access Time	t_{ACC}		—	—	370	—	—	240	—	—	190	ns

2. MPU WRITE TIMING

Item	Symbol	Test Condition	HD6340			HD63A40			HD63B40			Unit
			min	typ	max	min	typ	max	min	typ	max	
Enable Cycle Time	t_{cycE}	Fig. 2	1000	—	10000	666	—	10000	500	—	10000	ns
Enable "High" Pulse Width	PW_{EH}		450	—	9500	280	—	9500	220	—	9500	ns
Enable "Low" Pulse Width	PW_{EL}		430	—	9500	280	—	9500	210	—	9500	ns
Enable Rise and Fall Time	t_{Er}, t_{Ef}		—	—	25	—	—	25	—	—	20	ns
Address Set Up Time	t_{AS}		80	—	—	60	—	—	40	—	—	ns
Data Set Up Time	t_{DSW}		165	—	—	80	—	—	60	—	—	ns
Data Hold Time	t_{HW}		10	—	—	10	—	—	10	—	—	ns
Address Hold Time	t_{AH}		10	—	—	10	—	—	10	—	—	ns

3. TIMING OF PTM SIGNAL

Item		Symbol	Test Condition	HD6340		HD63A40		HD63B40		Unit
				min	max	min	max	min	max	
Input Rise and Fall Times	$\overline{C}$, $\overline{G}$, $\overline{RES}$	t_r, t_f	Fig. 3, Fig. 4	—	1000*	—	666*	—	500*	ns
Input "Low" Pulse Width	$\overline{C}$, $\overline{G}$, $\overline{RES}$	PW_L	Fig. 3 (Asynchronous Mode)	$t_{cycE}+t_{SU}+t_{HD}$	—	$t_{cycE}+t_{SU}+t_{HD}$	—	$t_{cycE}+t_{SU}+t_{HD}$	—	ns
Input "High" Pulse Width	$\overline{C}$, $\overline{G}$	PW_H	Fig. 4 (Asynchronous Mode)	$t_{cycE}+t_{SU}+t_{HD}$	—	$t_{cycE}+t_{SU}+t_{HD}$	—	$t_{cycE}+t_{SU}+t_{HD}$	—	ns
Input Setup Time	$\overline{C}$, $\overline{G}$, $\overline{RES}$	t_{SU}	Fig. 5 (Synchronous Mode)	200	—	120	—	75	—	ns
	$\overline{C_3}$ (÷8 Prescaler Mode)			200	—	170	—	170	—	
Input Hold Time	$\overline{C}$, $\overline{G}$, $\overline{RES}$	t_{HD}	Fig. 5 (Synchronous Mode)	50	—	50	—	50	—	ns
	$\overline{C_3}$ (÷8 Prescaler Mode)			50	—	50	—	50	—	
Input Pulse Width	$\overline{C_3}$ (÷8 Prescaler Mode)	PW_L, PW_H	(Asynchronous Mode)	120	—	80		60	—	ns
Output Delay Time	$O_1 \sim O_3$	t_{co}	Fig. 6	—	200	—	200	—	200	ns
Interrupt Release Time		t_{iR}	Fig. 7	—	1200	—	900	—	700	ns

* t_r, $t_f \leq t_{cycE}$

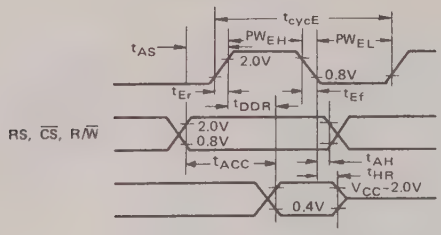


Figure 1 Bus Read Timing
(Read Information from PTM)

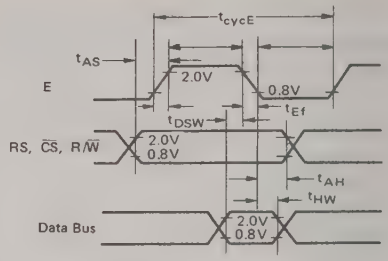


Figure 2 Bus Write Timing
(Write Information into PTM)

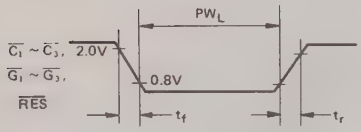


Figure 3 Input Pulse Width "Low"

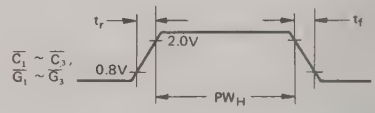


Figure 4 Input Pulse Width "High"

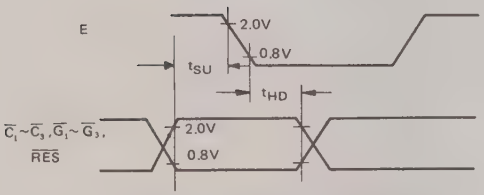


Figure 5 Input Setup and Hold Times

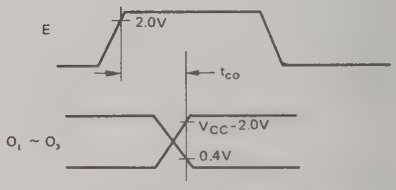


Figure 6 Output Delay

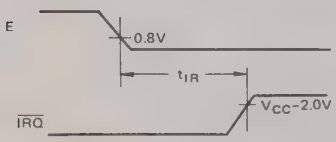


Figure 7 $\overline{IRQ}$ Release Time

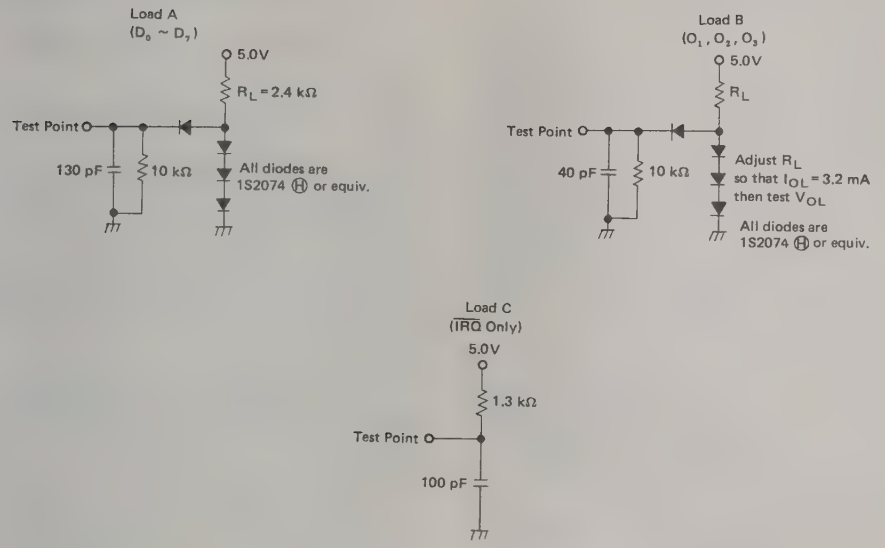


Figure 8 Test Loads

■ GENERAL DESCRIPTION

The HD6340 is part of the HMCS6800 microprocessor family and is fully bus compatible with HD6800 systems. The three timers in the HD6340 operate independently and in several distinct modes to fit a wide variety of measurement and synthesis applications.

The HD6340 is an integrated set of three distinct counter/timers. It consists of three 16-bit data latches, three 16-bit counters (clocked independently), and the comparison and enable circuitry necessary to implement various measurement and synthesis functions. In addition, it contains interrupt drivers to alert the processor that a particular function has been completed.

In a typical application, a timer will be loaded by first storing two bytes of data into an associated Counter Latch. This data is then transferred into the counter via a Counter initialization cycle. If the counter is enabled, the counter decrements on each subsequent clock period which may be an external clock, or Enable (E) until one of several predetermined conditions causes it to halt or recycle. The timers are thus programmable, cyclic in nature, controllable by external inputs or the MPU program, and accessible by the MPU at any time.

■ PTM INTERFACE SIGNALS FOR MPU

The Programmable Timer Module (PTM) interfaces to the HMCS6800 Bus with an eight-bit bidirectional data bus, two Chip Select lines, a Read/Write line, an Enable (System ϕ_2) line, an Interrupt Request line, an external Reset line, and three Register Select lines. These signals, in conjunction with the HD6800 VMA output, permit the MPU to control the PTM. VMA should be utilized in conjunction with an MPU address line into a Chip Select of the PTM, when the HD6800, HD6802 are used.

- **Bidirectional Data ($D_0 \sim D_7$)**

The bidirectional data lines (D_0~D_7) allow the transfer of data between the MPU and PTM. The data bus output drivers are three-state devices which remain in the high-impedance (off) state except when the MPU performs a PTM read operation (Read/Write and Enable lines "High" and PTM Chip Selects activated).

- **Chip Select ($\overline{CS_0}$, CS_1)**

These two signals are used to activate the Data Bus interface and allow transfer of data from the PTM. With $\overline{CS_0}$ = "Low" and CS_1 = "High", the device is selected and data transfer will occur.

- **Read/Write (R/$\overline{W}$)**

This signal is generated by the MPU to control the direction of data transfer on the Data Bus. With the PTM selected, a "Low" state on the PTM R/$\overline{W}$ line enables the input buffers and

data is transferred from the MPU to the PTM on the trailing edge of the Enable (System ϕ_2) signal. Alternately, (under the same conditions) R/$\overline{W}$ = "High" and Enable "High" allows data in the PTM to be read by the MPU.

- **Enable (E)**

This signal synchronizes data transfer between the MPU and the PTM. It also performs an equivalent synchronization function on the external clock, reset, and gate inputs of the PTM.

- **Interrupt Request ($\overline{IRQ}$)**

The active "Low" Interrupt Request signal is normally tied directly (or through priority interrupt circuitry) to the $\overline{IRQ}$ input of the MPU. This is an "open drain" output (no load device on the chip) which permits other similar interrupt request lines to be tied together in a wire-OR configuration.

The $\overline{IRQ}$ line is activated if, and only if, the Composite Interrupt Flag (Bit 7 of the Internal Status Register) is asserted. The conditions under which the $\overline{IRQ}$ line is activated are discussed in conjunction with the Status Register.

- **Reset ($\overline{RES}$)**

A "Low" level at this input is clocked into the PTM by the Enable (System ϕ_2) input. Two Enable pulses are required to synchronize and process the signal. The PTM then recognizes the active "Low" or inactive "High" on the third Enable pulse. If the $\overline{RES}$ signal is asynchronous, an additional Enable period is required if setup times are not met. The $\overline{RES}$ input must be stable "High"/"Low" for the minimum time stated in the AC Characteristics.

Recognition of a "Low" level at this input by the PTM causes the following action to occur:

a. All counter latches are preset to their maximal count values.
b. All Control Register bits are cleared with the exception of CR10 (internal reset bit) which is set.
c. All counters are preset to the contents of the latches.
d. All counter outputs are reset and all counter clocks are disabled.
e. All Status Register bits (interrupt flags) are cleared.

- **Register Select Lines (RS_0, RS_1, RS_2)**

These inputs are used in conjunction with the R/$\overline{W}$ line to select the internal registers, counters and latches as shown in Table 1.

It has been previously stated that the PTM is accessed via MPU Load and Store operations in much the same manner as a memory device. The instructions available with the HMCS6800 family of MPUs which perform operations directly on memory should not be used when the PTM is accessed. These instructions actually fetch a byte from memory, perform an operation, then restore it to the same address location. Since the PTM used the R/$\overline{W}$ line as an additional register select input, the modified data may not be restored to the same register if these instructions are used.

■ PTM ASYNCHRONOUS INPUT/OUTPUT SIGNALS

Each of the three timers within the PTM has external clock and gate inputs as well as a counter output line. The inputs are high impedance, TTL compatible lines and outputs are capable of driving two standard TTL loads.

- **Clock Inputs ($\overline{C_1}$, $\overline{C_2}$, $\overline{C_3}$)**

Input pins $\overline{C_1}$, $\overline{C_2}$, and $\overline{C_3}$ will accept asynchronous TTL voltage level signals to decrement Timers 1, 2, and 3, respectively. The "High" and "Low" levels of the external clocks must each be stable for at least one system clock period plus the sum

Table 1 Register Selection

Register Select Inputs *			Operations	
RS_2	RS_1	RS_0	R/$\overline{W}$ = "Low"	R/$\overline{W}$ = "High"
L	L	L	CR20 = "0" Write Control Register #3 CR20 = "1" Write Control Register #1	All bits "0"
L	L	H	Write Control Register #2	Read Status Register
L	H	L	Write MSB Buffer Register	Read Timer #1 Counter
L	H	H	Write Timer #1 Latches	Read LSB Buffer Register
H	L	L	Write MSB Buffer Register	Read Timer #2 Counter
H	L	H	Write Timer #2 Latches	Read LSB Buffer Register
H	H	L	Write MSB Buffer Register	Read Timer #3 Counter
H	H	H	Write Timer #3 Latches	Read LSB Buffer Register

* L; "Low" level, H; "High" level

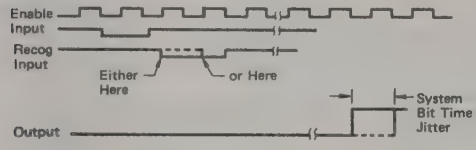
of the setup and hold times for the inputs. The asynchronous clock rate can vary from dc to the limit imposed by Enable (System ϕ_2) Setup, and Hold time.

The external clock inputs are clocked in by Enable (System ϕ_2) pulses. Three Enable periods are used to synchronize and process the external clock. The fourth Enable pulse decrements the internal counter. This does not affect the input frequency, it merely creates a delay between a clock input transition and internal recognition of that transition by the PTM. All references to $\overline{C}$ inputs in this document relate to internal recognition of the input transition. Note that a clock "High" or "Low" level which does not meet setup and hold time specifications may require an additional Enable pulse for recognition. When observing recurring events, a lack of synchronization will result in "jitter" being observed on the output of the PTM when using asynchronous clocks and gate input signals. There are two types of jitter. "System jitter" is the result of the input signals being out of synchronization with the Enable (System ϕ_2), permitting signals with marginal setup and hold time to be recognized by either the bit time nearest the input transition or the subsequent bit time.

"Input jitter" can be as great as the time between input signal negative going transitions plus the system jitter, if the first transition is recognized during one system cycle, and not recognized the next cycle, or vice versa.

External clock input $\overline{C_3}$ represents a special case when Timer

#3 is programmed to utilize its optional ÷8 prescaler mode. The maximum input frequency and allowable duty cycles for this case are specified under the AC Characteristics. The output of the ÷8 prescaler is treated in the same manner as the previously discussed clock inputs. That is, it is clocked into the counter by Enable pulses, is recognized on the fourth Enable pulse (provided setup and hold time requirements are met), and must produce an output pulse at least as wide as the sum of an Enable period, setup, and hold times.



- **Gate Inputs ($\overline{G_1}$, $\overline{G_2}$, $\overline{G_3}$)**

Input pins $\overline{G_1}$, $\overline{G_2}$, and $\overline{G_3}$ accept asynchronous TTL-compatible signals which are used as triggers or clock gating functions to Timers 1, 2, and 3, respectively. The gating inputs are clocked into the PTM by the Enable (System ϕ_2) signal in the same manner as the previously discussed clock inputs. That is, a $\overline{Gate}$ transition is recognized by the PTM on the fourth Enable pulse (provided setup and hold time requirements are met), and the "High" or "Low" levels of the $\overline{Gate}$ input must be stable for at least one system clock period plus the sum of setup and hold times. All references to $\overline{G}$ transition in this document relate to internal recognition of the input transition.

The $\overline{Gate}$ inputs of all timers directly affected the internal 16-bit counter. The operation of $\overline{G_3}$ is therefore independent of the ÷8 prescaler selection.

- **Timer Outputs (O_1, O_2, O_3)**

Timer outputs O_1, O_2, and O_3 are capable of driving up to two TTL loads and produce a defined output waveform for either Continuous or Single-Shot Timer modes. Output waveform definition is accomplished by selecting either Single 16-bit or Dual 8-bit operating modes. The single 16-bit mode will produce a square-wave output in the continuous timer mode and will produce a single pulse in the Single-Shot Timer mode. The Dual 8-bit mode will produce a variable duty cycle pulse in both the continuous and single shot Timer modes. "1" bit of each Control Register (CRX7) is used to enable the corresponding output. If this bit is cleared, the output will remain "Low" (V_{OL}) regardless of the operating mode.

If it is cleared while the output is high the output will go low during the first enable cycle following a write to the Control Register.

The Continuous and Single-Shot Timer Modes are the only ones for which output response is defined in this data sheet. Signals appear at the outputs (unless CRX7= "0") during Frequency and Pulse Width comparison modes, but the actual waveform is not predictable in typical applications.

■ CONTROL REGISTER

Each timer in the HD6340 has a corresponding write-only Control Register. Control Register #2 has a unique address space (RS0="High", RS1="Low", RS2="Low") and therefore may be written into at any time. The remaining Control Registers (#1 and#3) share the Address Space selected by a "Low" level on all Register Select inputs.

- **CR20**

The least-significant bit of Control Register #2 (CR20) is used as an additional addressing bit for Control Registers #1 and #3. Thus, with all Register selects and R/$\overline{W}$ inputs at "Low" level. Control Register #1 will be written into if CR20 is a logic "1". Under the same conditions, control Register #3 can also be written into after a $\overline{RES}$ "Low" condition has occurred, since all control register bits (except CR10) are cleared. Therefore, one may write in the sequence CR3, CR2, CR1.

- **CR10**

The least-significant bit of Control Register #1 is used as an internal Reset bit. When this bit is a logic "0", all timers are allowed to operate in the modes prescribed by the remaining bits of the control registers. Writing a "1" into CR10 causes all counters to be preset with the contents of the corresponding counter latches, all counter clocks to be disabled, and the timer outputs and interrupt flags (Status Register) to be reset. Counter Latches and Control Registers are undisturbed by an Internal Reset and may be written into regardless of the state of CR10.

- **CR30**

The least-significant bit of Control Register #3 is used as a selector for a ÷8 prescaler which is available with Timer #3 only. The prescaler, if selected, is effectively placed between the clock input circuitry and the input to Counter #3. It can therefore be used with either the internal clock (Enable) or an external clock source.

- **CRX1 ~ CRX7 (X=1~3)**

The functions depicted in the foregoing discussions are tabulated in Table 2 for ease of reference.

Control Register Bits CR10, CR20, and CR30 are unique in that each selects a different function. The remaining bits (1 through 7) of each Control Register select common functions, with a particular Control Register affecting only its corresponding timer.

- **CRX1**

Bit 1 of Control Register #1 (CR11) selects whether an internal or external clock source is to be used with Timer #1. Similarly, CR21 selects the clock source for Timer #2, and CR31 performs this function for Timer #3. The function of each bit of Control Register "X" can therefore be defined as shown in the remaining section of Table 2.

- **CRX2**

Control Register Bit 2 selects whether the binary information contained in the Counter Latches (and subsequently loaded into the counter) is to be treated as a single 16-bit word or two 8-bit bytes. In the single 16-bit Counter Mode (CRX2=0) the counter will decrement to zero after N + 1 enabled ($\overline{G}$="Low") clock periods, where N is defined as the 16-bit number in the Counter Latches. With CRX2 = 1, a similar Time Out will occur after (L + 1)·(M + 1) enabled clock periods, where L and M, respectively, refer to the LSB and MSB bytes in the Counter Latches.

- **CRX3 ~ CRX7**

Control Register Bits 3, 4, and 5 are explained in detail in the Timer Operating Mode section. Bit 6 is an interrupt mask bit which will be explained more fully in conjunction with the Status Register, and bit 7 is used to enable the corresponding Timer Output. A summary of the control register programming modes is shown in Table 3.

■ STATUS REGISTER/INTERRUPT FLAGS

The HD6340 has an internal Read-Only Status Register which contains four Interrupt Flags. (The remaining four bits of the register are not used, and default to "0"s when being read.) Bits 0, 1, and 2 are assigned to Timers 1, 2, and 3, respectively, as individual flag bits, while Bit 7 is a Composite Interrupt Flag. This flag bit will be asserted if any of the individual flag bits is

Table 2 Control Register Bits

CONTROL REGISTER #1		CONTROL REGISTER #2		CONTROL REGISTER #3	
CR10	Internal Reset Bit	CR20	Control Register Address Bit	CR30	Timer #3 Clock Control
"0" All timers allowed to operate "1" All timers held in preset state		"0" CR #3 may be written "1" CR #1 may be written		"0" T3 Clock is not prescaled "1" T3 Clock is prescaled by ÷ 8	
CRX1*		Timer #X Clock Source			
"0"		TX uses external clock source on $\overline{CX}$ input			
"1"		TX uses Enable clock			
CRX2		Timer #X Counting Mode Control			
"0"		TX configured for normal (16-bit) counting mode			
"1"		TX configured for dual 8-bit counting mode			
CRX3 CRX4 CRX5		Timer #X Counter Mode and Interrupt Control (See Table 3)			
CRX6		Timer #X Interrupt Enable			
"0"		Interrupt Flag masked on $\overline{IRQ}$			
"1"		Interrupt Flag enabled to $\overline{IRQ}$			
CRX7		Timer #X Counter Output Enable			
"0"		TX Output masked on output OX			
"1"		TX Output enabled on output OX			

* Control Register for Timer 1, 2, or 3, Bit 1.

set while Bit 6 of the corresponding Control Register is at a logic "1". The conditions for asserting the Composite Interrupt Flag bit can therefore be expressed as:

$$INT = I_1 \cdot CR16 + I_2 \cdot CR26 + I_3 \cdot CR36$$

where INT = Composite Interrupt Flag (Bit 7)
I_1 = Timer #1 Interrupt Flag (Bit 0)
I_2 = Timer #2 Interrupt Flag (Bit 1)
I_3 = Timer #3 Interrupt Flag (Bit 2)

STATUS REGISTER

7	6	5	4	3	2	1	0
INT					I_3	I_2	I_1

An interrupt flag is cleared by a Timer Reset condition, i.e., External $\overline{RES}$ = "Low" or Internal Reset Bit (CR10) = "1". It will also be cleared by a Read Timer Counter Command provided that the Status Register has previously been read while the interrupt flag was set. This condition on the Read Status Register – Read Timer Counter (RS–RT) sequence is designed to prevent missing interrupts which might occur after the status register is read, but prior to reading the Timer Counter.

An Individual Interrupt Flag is also cleared by a Write Timer Latches (W) command or a Counter Initialization (CI) sequence, provided that W or CI affects the Timer corresponding to the individual Interrupt Flag.

■ COUNTER LATCH INITIALIZATION

Each of the three independent timers consists of a 16-bit addressable counter and 16 bits of addressable latches. The counters are preset to the binary numbers stored in the latches. Counter initialization results in the transfer of the latch contents to the counter. See notes in Table 5 regarding the binary number N, L, or M placed into the Latches and their relationship to the output waveforms and counter Time-Outs.

Since the PTM data bus is 8-bits wide and the counters are 16-bits wide, a temporary register (MSB Buffer Register) is provided. This "write only" register is for the Most Significant Byte of the desired latch data. Three addresses are provided for the MSB Buffer Register (as indicated in Table 1), but they all lead to the same Buffer. Data from the MSB Buffer will automatically be transferred into the Most Significant Byte of Timer #X when a Write Timer #X Latches Command is performed. So it can be seen that the HD6340 has been designed to allow transfer of two bytes of data into the counter latches provided that the MSB is transferred first.

In many applications, the source of the data will be as HMCS6800 MPU. It should be noted that the 16-bit store operations of the HMCS6800 microprocessors (STS and STX etc.) transfer data in the order required by the PTM. A Store Index Register Instruction, for example, results in the MSB of the X register being transferred to the selected address, then the LSB of the X register being written into the next higher location. Thus, either the index register or stack pointer may be transfered directly into a selected counter latch with a single instruction.

A logic "Low" at the $\overline{RES}$ input also initializes the counter latches. In this case, all latches will assume a maximum count of $(65,536)_{10}$. It is important to note that an Internal Reset (Bit 0 of Control Register 1 Set) has no effect on the counter latches.

■ COUNTER INITIALIZATION

Counter Initialization is defined as the transfer of data from the latches to the counter with subsequent clearing of the Individual Interrupt Flag associated with the counter. Counter Initialization always occurs when a reset condition ($\overline{RES}$ = "Low" or CR10 = "1") is recognized. It can also occur – depending on Timer Mode – with a Write Timer Latches command or recognition of a negative transition of the $\overline{Gate}$ input.

Counter recycling or re-initialization occurs when a negative transition of the clock input is recognized after the counter has reached an all-zero state. In this case, data is transferred from the Latches to the Counter.

■ TIMER OPERATING MODES

The HD6340 has been designed to operate effectively in a wide variety of applications. This is accomplished by using three bits of each control register (CRX3, CRX4, and CRX5) to

defined different operating modes of the Timers. These modes are divided into Wave Synthesis and Wave Measurement modes, and outlined in Table 4.

Table 4 Operating Modes

Control Register				
CRX3	CRX4	CRX5	Timer Operating Mode	
0	*	0	Continuous	Wave Synthesis
0	*	1	Single-Shot	
1	0	*	Frequency Comparison	Wave Measurement
1	1	*	Pulse Width Comparison	

* Defines Additional Timer Functions.

One of the WAVE SYNTHESIS modes is the Continuous Operating mode, which is useful for cyclic wave generation. Either symmetrical or variable duty-cycle waves can be generated in this mode. The other wave synthesis mode, the Single-Shot mode, is similar in use to the Continuous operating mode, however, a single pulse is generated, with a programmable preset width.

The WAVE MEASUREMENT modes include the Frequency Comparison and Pulse Width Comparison modes which are used to measure cyclic and singular pulse widths, respectively.

In addition to the four timer modes in Table 4, the remaining control register bit is used to modify counter initialization and enabling or interrupt conditions.

■ WAVE SYNTHESIS MODES

• Continuous Operating Mode (Table 5)

The continuous mode will synthesize a continuous wave with

Table 3 Control Register Programming

7	6	5	4	3	2	1	0	Register 1	Register 2	Register 3
X	X	X	X	X	X	X	↕	"0" All Timers Operate	Reg #3 May Be Written	T3 Clk ÷ 1
								"1" All Timers Preset	Reg #1 May Be Written	T3 Clk ÷ 8

7	6	5	4	3	2	1	0	
X	X	X	X	X	X	↕	X	"0" External Clock ($\overline{CX}$ Input) "1" Internal Clock (Enable)

7	6	5	4	3	2	1	0	
X	X	X	X	X	↕	X	X	"0" Normal (16-Bit) Count Mode "1" Dual 8-Bit Count Mode

7	6	5	4	3	2	1	0	
X	X	0	0	0	X	X	X	Continuous Operating Mode: $\overline{Gate}$ ↓ or Write to Latches or Reset Causes Counter Initialization
X	X	0	0	1	X	X	X	Frequency Comparison Mode: Interrupt if $\overline{Gate}$ ⊓ is < Counter Time Out
X	X	0	1	0	X	X	X	Continuous Operating Mode: $\overline{Gate}$ ↓ or Reset Causes Counter Initialization
X	X	0	1	1	X	X	X	Pulse Width Comparison Mode: Interrupt if $\overline{Gate}$ ⊓ is < Counter Time Out
1	X	1	0	0	X	X	X	Single Shot Mode: $\overline{Gate}$ ↓ or Write to Latches or Reset Causes Counter Initialization
X	X	1	0	1	X	X	X	Frequency Comparison Mode: Interrupt If $\overline{Gate}$ ⊓ is > Counter Time Out
1	X	1	1	0	X	X	X	Single Shot Mode: $\overline{Gate}$ ↓ or Reset Causes Counter Initialization
X	X	1	1	1	X	X	X	Pulse Width Comparison Mode: Interrupt If $\overline{Gate}$ ⊓ is > Counter Time Out

7	6	5	4	3	2	1	0	
X	↕	X	X	X	X	X	X	"0" Interrupt Flag Masked ($\overline{IRQ}$) "1" Interrupt Flag Enabled ($\overline{IRQ}$)

7	6	5	4	3	2	1	0	
↕	X	X	X	X	X	X	X	"0" Timer Output Masked "1" Timer Output Enable

(NOTE) Reset is Hardware or Software Reset ($\overline{RES}$ = "Low" or CR10 = "1").

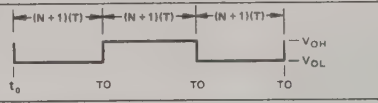
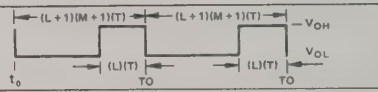
a period proportional to the preset number in the particular timer latches.

Any of the timers in the PTM may be programmed to operate in a continuous mode by writing "0"s into bits 3 and 5 of the corresponding control register. Assuming that the timer output is enabled (CRX7 = "1"), either a square wave or a variable duty cycle waveform will be generated at the Timer Output, OX. The type of output is selected via Control Register Bit 2.

Either a Timer Reset (CR10 = "1" or External $\overline{RES}$ = "Low") condition or internal recognition of a negative transition of the $\overline{Gate}$ input results in Counter Initialization. A Write Timer Latches command can be selected as a Counter Initialization signal by clearing CRX4.

The counter is enabled by an absence of a Timer Reset condition and a "Low" level at the $\overline{Gate}$ input. In the 16-bit mode, the counter will decrement on the first clock cycle during or after the counter initialization cycle. It continues to decrement on each clock signal so long as $\overline{G}$ remains "Low" and no reset condition exists. A Counter Time Out (the first clock after all

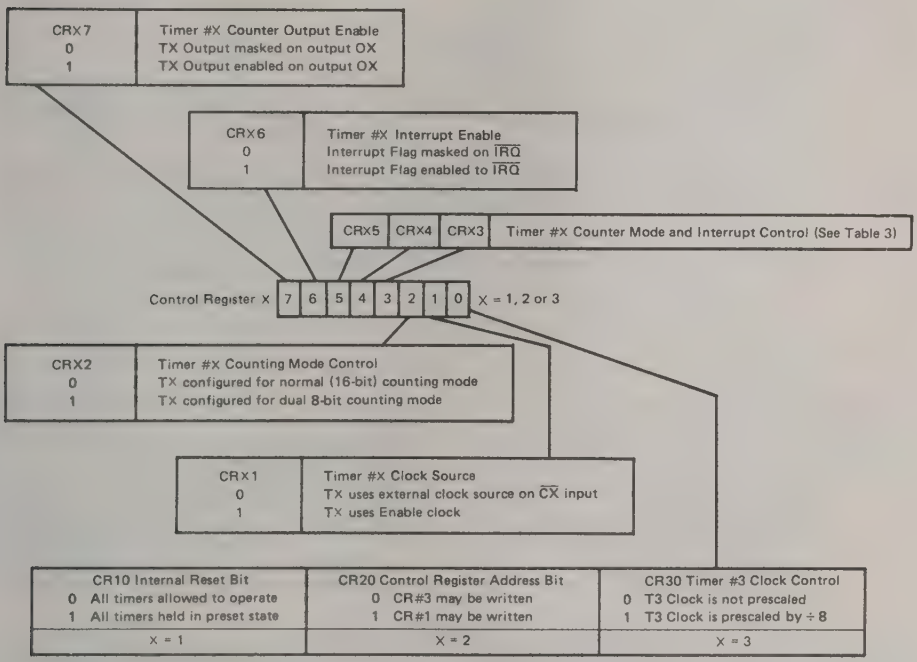
Table 5 Continuous Operating Modes

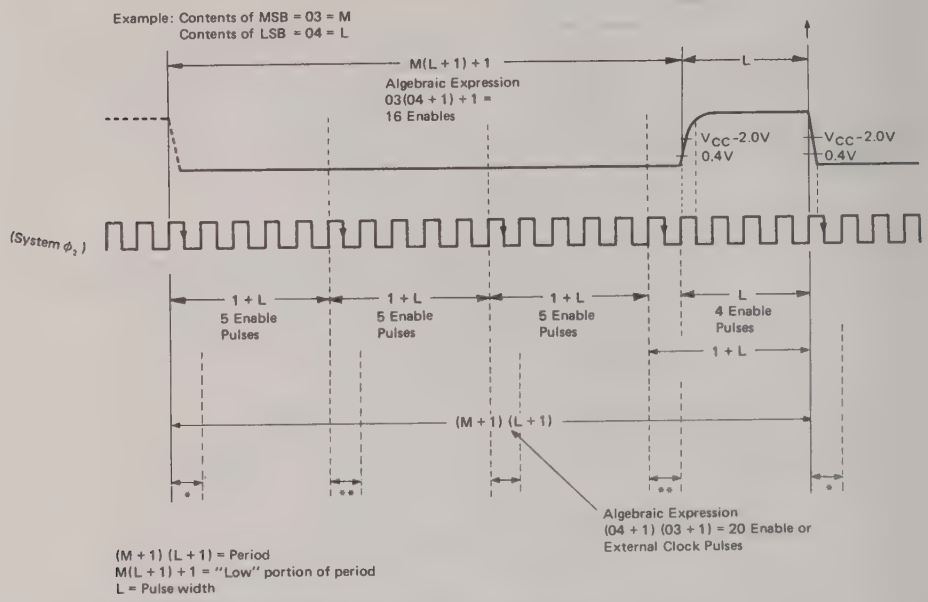
CONTINUOUS MODE (CRX3 = "0", CRX5 = "0")			
Control Register		Initialization/Output Waveforms	
CRX2	CRX4	Counter Initialization	*Timer Output (OX) (CRX7 = "1")
0	0	$\overline{G}\downarrow$+W+R	
0	1	$\overline{G}\downarrow$+R	
1	0	$\overline{G}\downarrow$+W+R	
1	1	$\overline{G}\downarrow$+R	

$\overline{G}\downarrow$ = Negative transition of $\overline{Gate}$ input.
W = Write Timer Latches Command.
R = Timer Reset (CR10 = "1" or External $\overline{RES}$ = "Low")
N = 16-Bit Number in Counter Latch.
L = 8-Bit Number in LSB Counter Latch.
M = 8-Bit Number in MSB Counter Latch.
T = Clock Input Negative Transitions to Counter.
t_0 = Counter Initialization Cycle.
TO = Counter Time Out (All Zero Condition).

* All time intervals shown above assume the $\overline{Gate}$ ($\overline{G}$) and $\overline{Clock}$ ($\overline{C}$) signals are synchronized to Enable (System ϕ_2) with the specified setup and hold time requirements.

Control Register Bits





* Preset LSB and MSB to Respective Latches on the negative transition of the E.
** Preset LSB to LSB Latches and Decrement MSB by one on the negative transition of the E.

Figure 9 Timer Output Waveform Example
(Continuous Dual 8-Bit Mode using Internal Enable)

counter bits = "0") results in the Individual Interrupt Flag being set and re-initialization of the counter.

In the dual 8-bit mode (CRX2= "1") [Refer to the example in Fig. 9] the MSB decrements once for every full countdown of the LSB + 1. When the LSB = "0", the MSB is unchanged; on the next clock pulse the LSB is reset to the count in the LSB Latches and the MSB is decremented by 1 (one). The output, if enabled, remains "Low" during and after initialization and will remain "Low" until the counter MSB is all "0"s. The output will go "High" at the beginning of the next clock pulse. The output remains "High" until both the LSB and MSB of the counter are all "0"s. At the beginning of the next clock pulse the defined Time Out (TO) will occur and the output will go "Low". In the Dual 8-bit mode the period of the output of the example in Fig. 9 would span 20 clock pulses as opposed to the 1546 clock pulses using the Normal 16-bit mode.

A special time-out condition exists for the dual 8-bit mode (CRX2 = "1") if L = "0". In this case, the counter will revert to a mode similar to the single 16-bit mode, except Time Out occurs after M+1 clock pulses. The output, if enabled, goes "Low" during the Counter Initialization cycle and reverses state at each Time Out. The counter remains cyclical (is re-initialized at each Time Out) and the Individual Interrupt Flag is set when Time Out occurs. If M = L = "0", the internal counters do not change, but the output toggles at a rate of 1/2 the clock frequency.

The discussion of the Continuous Mode has assumed that the application requires an output signal. It should be noted that the Timer operates in the same manner with the output disabled (CRX7 = "0"). A Read Timer Counter command is valid regardless of the state of CRX7.

- **Single-Shot Timer Mode**

This mode is identical to the Continuous Mode with three exceptions. The first of these is obvious from the name – the output returns to a "Low" level after the initial Time Out and remains "Low" until another Counter Initialization cycle occurs. The waveforms available are shown in Table 6.

As indicated in Table 6, the internal counting mechanism remains cyclical in the Single-Shot Mode. Each Time Out of the counter results in the setting of an Individual Interrupt Flag and re-initialization of the counter.

The second major difference between the Single-Shot and Continuous modes is that the internal counter enable is not dependent on the $\overline{\text{Gate}}$ input level remaining in the "Low" state for the Single-Shot mode.

Another special condition is introduced in the Single-Shot mode. If L = M = "0"(Dual 8-bit) or N = "0" (Single 16-bit), the output goes "Low" on the first clock received during or after Counter Initialization. The output remains "Low" until the Operating Mode is changed or nonzero data is written into the Counter Latches. Time Outs continue to occur at the end of each clock period.

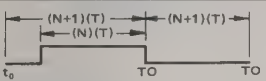
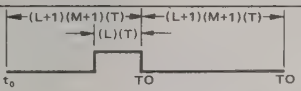
The three differences between Single-Shot and Continuous Timer Modes can be summarized as attributes of the Single-Shot

mode:

1. Output is enabled for only one pulse until it is reinitialized.
2. Counter Enable is independent of $\overline{Gate}$.
3. L = M = "0" or N = "0" disables output.

Aside from these differences, the two modes are identical.

Table 6 Single-Shot Operating Modes

Single-Shot Mode (CRX3 = "0", CRX7 = "1", CRX5 = "1")			
Control Register		Initialization/Output Waveforms	
CRX2	CRX4	Counter Initialization	Timer Output (OX)
0	0	$\overline{G}\downarrow$+W+R	
0	1	$\overline{G}\downarrow$+R	
1	0	$\overline{G}_*$+W+R	
1	1	$\overline{G}\downarrow$+R	

Symbols are as defined in Table 5.

■ WAVE MEASUREMENT MODES

The Wave Measurement Modes are the Frequency (period) Measurement and Pulse Width Comparison Modes, and are provided for those applications which require more flexibility of interrupt generation and Counter Initialization. Individual Interrupt Flags are set in these modes as a function of both Counter Time Out and transitions of the $\overline{Gate}$ input. Counter Initialization is also affected by Interrupt Flag status.

A timer's output is normally not used in a Wave Measurement mode, but it is defined. If the output is enabled, it will operate as follows. During the period between reinitialization of the timer and the first Time Out, the output will be a logical zero. If the first Time Out is completed (regardless of its method of generation), the output will go "High". If further TO's occur, the output will change state at each completion of a Time-Out.

The counter does operate in either Single 16-bit or Dual 8-bit modes as programmed by CRX2. Other features of the Wave Measurement Modes are outlined in Table 7.

Table 7 Wave Measurement Modes

CRX3 = "1"			
CRX4	CRX5	Application	Condition for Setting Individual Interrupt Flag
0	0	Frequency Comparison	Interrupt Generated if $\overline{Gate}$ Input Period (1/F) is less than Counter Time Out (TO)
0	1	Frequency Comparison	Interrupt Generated if $\overline{Gate}$ Input Period (1/F) is greater than Counter Time Out (TO)
1	0	Pulse Width Comparison	Interrupt Generated if $\overline{Gate}$ Input "Down Time" is less than Counter Time Out (TO)
1	1	Pulse Width Comparison	Interrupt Generated if $\overline{Gate}$ Input "Down Time" is greater than Counter Time Out (TO)

• Frequency Comparison or Period Measurement Mode (CRX3 = "1", CRX4 = "0")

The Frequency Comparison Mode with CRX5 = "1" is straightforward. If Time Out occurs prior to the first negative transition of the $\overline{Gate}$ input after a Counter Initialization cycle, an Individual Interrupt Flag is set. The counter is disabled, and a Counter Initialization cycle cannot begin until the interrupt flag is cleared and a negative transition on $\overline{G}$ is detected.

If CRX5 = "0", as shown in Table 7 and Table 8, an interrupt is generated if $\overline{Gate}$ input returns "Low" prior to a Time Out. If Counter Time-Out occurs first, the counter is recycled and continues to decrement. A bit is set within the timer on the initial Time Out which precludes further individual interrupt generation until a new Counter Initialization cycle has been completed. When this internal bit is set, a negative transition of the $\overline{Gate}$ input starts a new Counter Initialization cycle. (The condition of $\overline{G}\downarrow \cdot \overline{I} \cdot TO$ is satisfied, since a Time Out has occurred and no individual Interrupt has been generated.)

Any of the timers within the PTM may be programmed to compare the period of a pulse (giving the frequency after calculations) at the Gate input with the time period requested for Counter Time-Out. A negative transition of the $\overline{Gate}$ input enables the counter and starts a Counter Initialization cycle – provided that other conditions as noted in Table 8 are satisfied. The counter decrements on each clock signal recognized during or after Counter Initialization until an Interrupt is generated, a Write Timer Latches command is issued, or a Timer Reset condition occurs. It can be seen from Table 8 that an interrupt condition will be generated if CRX5 = "0" and the period of the pulse (single pulse or measured separately repetitive pulses) at the $\overline{Gate}$ input is less than the Counter Time Out period. If CRX5 = "1", an interrupt is generated if the reverse is true.

Assume now with CRX5 = "1" that a Counter Initialization has occurred and that the $\overline{Gate}$ input has returned "Low" prior to Counter Time Out. Since there is no Individual Interrupt Flag generated, this automatically starts a new Counter Initialization Cycle. The process will continue with frequency comparison being performed on each $\overline{Gate}$ input cycle until the mode is changed, or a cycle is determined to be above the predetermined limit.

- **Pulse Width Comparison Mode (CRX3 = "1", CRX4 = "1")**

This mode is similar to the Frequency Comparison Mode except for a positive, rather than negative, transition of the $\overline{Gate}$ input terminates the count. With CRX5 = "0", an Individual Interrupt Flag will be generated if the "Low" level pulse applied to the $\overline{Gate}$ input is less than the time period required for Counter Time Out. With CRX5 = "1", the interrupt is generated when the reverse condition is true.

As can be seen in Table 9, a positive transition of the $\overline{Gate}$ input disables the counter. With CRX5 = "0", it is therefore possible to directly obtain the width of any pulse causing an interrupt. Similar data for other Time Interval Modes and conditions can be obtained, if two sections of the PTM are dedicated to the purpose.

Table 8 Frequency Comparison Mode

CRX3 = "1", CRX4 = "0"				
Control Reg Bit 5 (CRX5)	Counter Initialization	Counter Enable Flip-Flop Set (CE)	Counter Enable Flip-Flop Reset (CE)	Interrupt Flag Set (I)
0	$\overline{G}\downarrow \cdot \overline{I} \cdot (\overline{CE}+TO)+R$	$\overline{G}\downarrow \cdot \overline{W} \cdot \overline{R} \cdot \overline{I}$	W+R+I	$\overline{G}\downarrow$ Before TO
1	$\overline{G}\downarrow \cdot \overline{I}+R$	$\overline{G}\downarrow \cdot \overline{W} \cdot \overline{R} \cdot \overline{I}$	W+R+I	TO Before $\overline{G}\downarrow$

I represents the interrupt for a given timer.

Table 9 Pulse Width Comparison Mode

CRX3 = "1", CRX4 = "1"				
Control Reg Bit 5 (CRX5)	Counter Initialization	Counter Enable Flip-Flop Set (CE)	Counter Enable Flip-Flop Reset (CE)	Interrupt Flag Set (I)
0	$\overline{G}\downarrow \cdot \overline{I}+R$	$\overline{G}\downarrow \cdot \overline{W} \cdot \overline{R} \cdot \overline{I}$	W+R+I+G	$\overline{G}\uparrow$ Before TO
1	$\overline{G}\downarrow \cdot \overline{I}+R$	$\overline{G}\downarrow \cdot \overline{W} \cdot \overline{R} \cdot \overline{I}$	W+R+I+G	TO Before $\overline{G}\uparrow$

G = Level sensitive recognition of $\overline{Gate}$ input.

■ NOTE FOR USE

Input signal, which is not necessary for user's application, should be used fixed to "High" or "Low" level. This is applicable to the following signal pins.

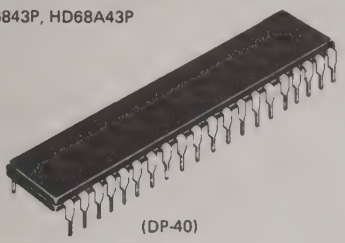
$\overline{C_1}$, $\overline{C_2}$, $\overline{C_3}$, $\overline{G_1}$, $\overline{G_2}$, $\overline{G_3}$

HD6843, HD68A43
FDC (Floppy Disk Controller)

The HD6843 Floppy Disk Controller performs the complex MPU/Floppy interface function. The FDC was designed to optimize the balance between the "Hardware/Software" in order to achieve integration of all key functions and maintain flexibility.

The FDC can interface a wide range of drives with a minimum of external hardware. Multiple drives can be controlled with the addition of external multiplexing rather than additional FDC's.

HD6843P, HD68A43P

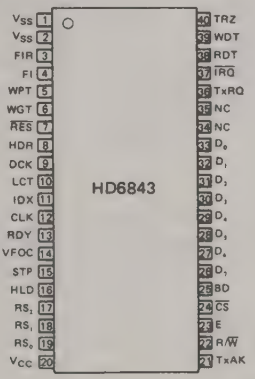


(DP-40)

■ FEATURES

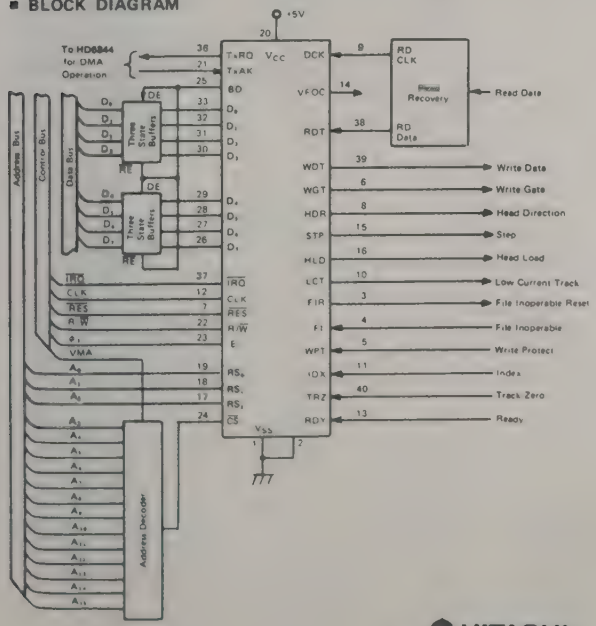
- Format compatible with IBM3740
- User Programmable read/write format
- Ten powerful macro-commands
- Macro End Interrupt allows parallel processing of MPU and FDC
- Controls multiple Floppies with external multiplexing
- Direct interface with HMCS6800
- Programmable seek and settling times enable operation with a wide range of Floppy drives
- Offers both Programmed Controlled I/O (PCIO) and DMA data transfer mode
- Free-Format read or write
- Single 5-volt power supply
- All registers directly accessible
- Compatible with MC6843

■ PIN ARRANGEMENT



(Top View)

■ BLOCK DIAGRAM



■ ABSOLUTE MAXIMUM RATINGS

Item	Symbol	Value	Unit
Supply Voltage	V_{CC}*	-0.3 ~ +7.0	V
Input Voltage	V_{in}*	-0.3 ~ +7.0	V
Operating Temperature	T_{opr}	0 ~ +75	°C
Storage Temperature	T_{stg}	-55 ~ +150	°C

* With respect to V_{SS} (SYSTEM GND)

(NOTE) Permanent LSI damage may occur if maximum ratings are exceeded. Normal operation should be under recommended operating conditions. If these conditions are exceeded, it could affect reliability of LSI.

■ RECOMMENDED OPERATING CONDITIONS

Item	Symbol	min.	typ.	max.	Unit
Supply Voltage	V_{CC}*	4.75	5.0	5.25	V
Input "High" Voltage	V_{IH}*	2.0	–	V_{CC}	V
Input "Low" Voltage	V_{IL}*	-0.3	–	0.8	V
Operating Temperature	T_{opr}	0	25	75	°C

* With respect to V_{SS} (SYSTEM GND)

■ ELECTRICAL CHARACTERISTICS

● DC CHARACTERISTICS (V_{CC}=5V±5%, V_{SS}=0V, Ta = 0 ~ +75°C, unless otherwise noted.)

Item		Symbol	Test Condition	min.	typ.*	max.	Unit
Input "High" Voltage		V_{IH}		2.0	–	V_{CC}	V
Input "Low" Voltage		V_{IL}		-0.3	–	0.8	V
Input Leakage Current		I_{in}	V_{in}=0~5.25V	–	1.0	2.5	μA
Output "High" Voltage		V_{OH}	I_{OH}=-205μA (D_0~D_7) I_{OH}=-100μA (Others)	2.4	–	–	V
Output "Low" Voltage		V_{OL}	I_{OL}=3.2mA ($\overline{IRQ}$) I_{OL}=1.6mA (Others)	–	–	0.4	V
Three-state (off-state) Leakage Current		I_{TSI}	V_{in}=0.4~2.4V	–	2.0	10	μA
Output Leakage (off-state) Current ($\overline{IRQ}$)		I_{LOH}	V_{OH}=2.4V	–	1.0	10	μA
Power Dissipation		P_D		–	600	1000	mW
Input Capacitance	D_0~D_7	C_{in}	V_{in}=0V, T_a=25°C, f=1 MHz	–	–	12.5	pF
	Other inputs			–	–	10	pF
Output Capacitance		C_{out}	V_{in}=0V, T_a=25°C, f=1 MHz	–	–	10	pF

* V_{CC} = 5V, Ta = 25°C

● AC CHARACTERISTICS (V_{CC}=5V±5%, V_{SS}=0V, Ta = 0 ~ +75°C, unless otherwise noted.)

Item	Symbol	Test Condition	HD6843			HD68A43			Unit
			min.	typ.	max.	min.	typ.	max.	
CLK Cycle Time	t_{cycC}	Figure 1	–	1.0	–	–	1.0	–	μs
CLK Pulse Width, "High"	PW_{HC}	Figure 1	0.4	–	–	0.4	–	–	μs
CLK Pulse Width, "Low"	PW_{LC}	Figure 1	0.35	–	–	0.35	–	–	μs
Rise and Fall Time of CLK	t_{Cr}, t_{Cf}	Figure 1	–	–	25	–	–	25	ns
DCK Cycle Time	t_{cycD}	Figure 2	2.6	4.0	–	2.6	4.0	–	μs
DCK Pulse Width, "High"	PW_{HD}	Figure 2	1.3	1.95	–	1.3	1.95	–	μs
DCK Pulse Width, "Low"	PW_{LD}	Figure 2	1.3	1.95	–	1.3	1.95	–	μs
Rise and Fall Time of DCK	t_{Dr}, t_{Df}	Figure 2	–	–	25	–	–	25	ns
RDT Width, "High"	t_{RDH}	Figure 2	1.0	–	–	1.0	–	–	μs
RDT Width, "Low"	t_{RDL}	Figure 2	1.0	–	–	1.0	–	–	μs
RDT~DCK Delay Time 1	t_{RDD1}	Figure 2	0.15	–	1.70	0.15	–	1.70	μs
RDT~DCK Delay Time 2	t_{RDD2}	Figure 2	0.15	–	1.70	0.15	–	1.70	μs
IDX Pulse Width, "High"	PW_{IDX}	Figure 3	20.0	–	–	20.0	–	–	μs
FIR Delay Time	t_{FIRD}	Figure 4	–	–	450	–	–	450	ns
FIR Pulse Width, "High"	PW_{FIR}	Figure 4	200	–	–	200	–	–	ns
WDT Pulse Width, "High"	PW_{WD}	Figure 7	–	1.0	–	–	1.0	–	μs
WDT Cycle Time	t_{cycW}	Figure 7	–	2.0	–	–	2.0	–	μs
STP Pulse Width, "High"	PW_{STP}	Figure 5	–	32	–	–	32	–	μs
STP Cycle Time	t_{cycS} *	Figure 5	1	–	15	1	–	15	ms
HLD Delay Time (HLD~STP)	t_{HLDD} *	Figure 5	1	–	15	1	–	15	ms
HDR Set Up Time	t_{HDRS}	Figure 5	0	–	–	0	–	–	ns
HDR Hold Time	t_{HDRH}	Figure 5	32	–	–	32	–	–	μs
TxAK Set Up Time	t_{AS3}	Figure 10, 11	140	–	–	140	–	–	ns
TxAK Hold Time	t_{AH3}	Figure 10, 11	10	–	–	10	–	–	ns
TxRQ Release Time	t_{TR}	Figure 10, 11	–	–	450	–	–	240	ns
$\overline{IRQ}$ Release Time	t_{IR}	Figure 6	–	–	1.2	–	–	1.2	μs

* Cycle Time of STP and HLD Delay Time change according to the program.

● BUS TIMING CHARACTERISTICS (V_{CC}=5V±5%, V_{SS}=0V, Ta = 0 ~ +75°C, unless otherwise noted.)

1 READ OPERATION SEQUENCE

Item	Symbol	Test Condition	HD6843			HD68A43			Unit
			min.	typ.	max.	min.	typ.	max.	
Enable Cycle Time	t_{cycE}	Figure 8, 10	1.0	–	–	0.666	–	–	μs
Enable Pulse Width, "High"	PW_{EH}	Figure 8, 10	0.4	–	–	0.23	–	–	μs
Enable Pulse Width, "Low"	PW_{EL}	Figure 8, 10	0.4	–	–	0.23	–	–	μs
Rise and Fall Time of Enable Input	t_{Er}, t_{Ef}	Figure 8, 10	–	–	25	–	–	25	ns
Address Set Up Time	t_{AS}	Figure 8, 10	140	–	–	140	–	–	ns
Data Delay Time	t_{DDR}	Figure 8, 10	–	–	225	–	–	200	ns
Data Access Time	t_{ACC}	Figure 8, 10	–	–	365	–	–	340	ns
Data Hold Time	t_{H}	Figure 8, 10	10	–	–	10	–	–	ns
Address Hold Time	t_{AH}	Figure 8, 10	10	–	–	10	–	–	ns
Bus Direction Delay Time	t_{DBD}	Figure 8, 10	–	–	400	–	–	400	ns

2 WRITE OPERATION SEQUENCE

Item	Symbol	Test Condition	HD6843			HD68A43			Unit
			min.	typ.	max.	min.	typ.	max.	
Enable Cycle Time	t_{cycE}	Figure 9, 11	1.0	–	–	0.666	–	–	μs
Enable Pulse Width, "High"	PW_{EH}	Figure 9, 11	0.4	–	–	0.23	–	–	μs
Enable Pulse Width, "Low"	PW_{EL}	Figure 9, 11	0.4	–	–	0.23	–	–	μs
Rise and Fall Time of Enable Input	t_{Er}, t_{Ef}	Figure 9, 11	–	–	25	–	–	25	ns
Address Set Up Time	t_{AS}	Figure 9, 11	140	–	–	140	–	–	ns
Data Set Up Time	t_{DSW}	Figure 9, 11	100	–	–	60	–	–	ns
Data Hold Time	t_H	Figure 9, 11	10	–	–	10	–	–	ns
Address Hold Time	t_{AH}	Figure 9, 11	10	–	–	10	–	–	ns
Bus Direction Delay Time	t_{DBD}	Figure 9, 11	–	–	400	–	–	400	ns

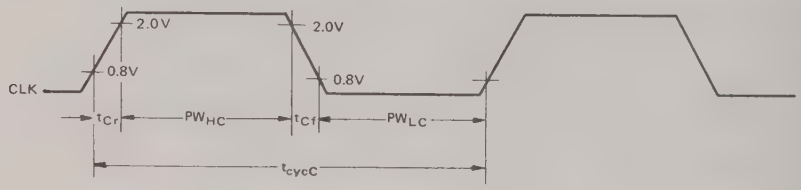


Figure 1 CLK Waveform

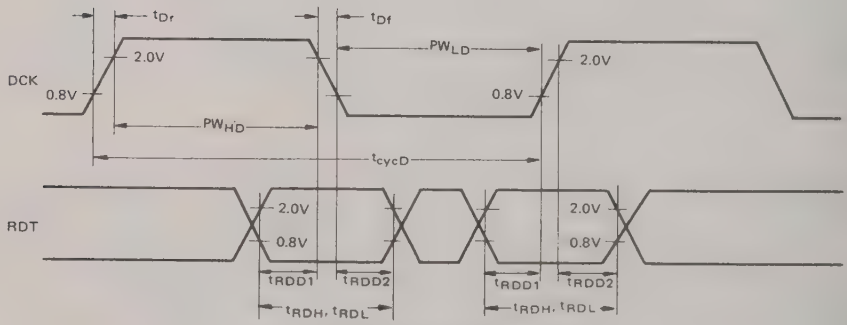


Figure 2 DCK, RDT Timing

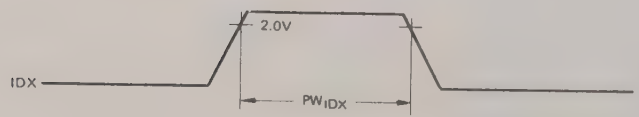


Figure 3 IDX Waveform

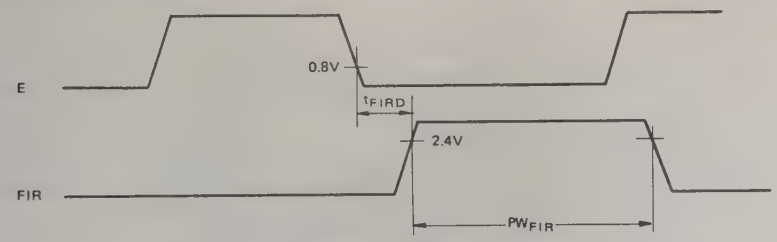


Figure 4 FIR Timing

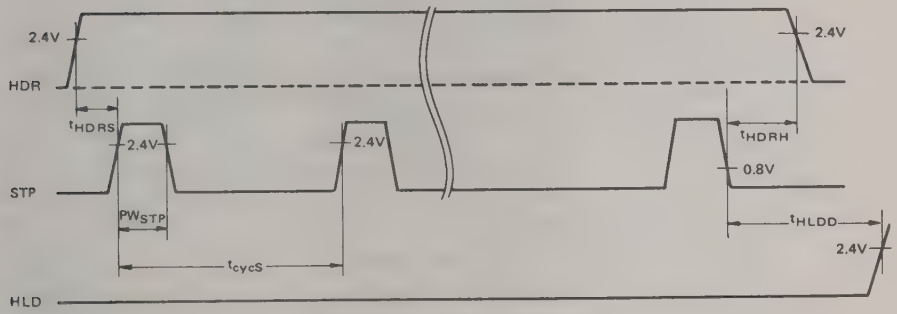


Figure 5 Seek Operation Sequence

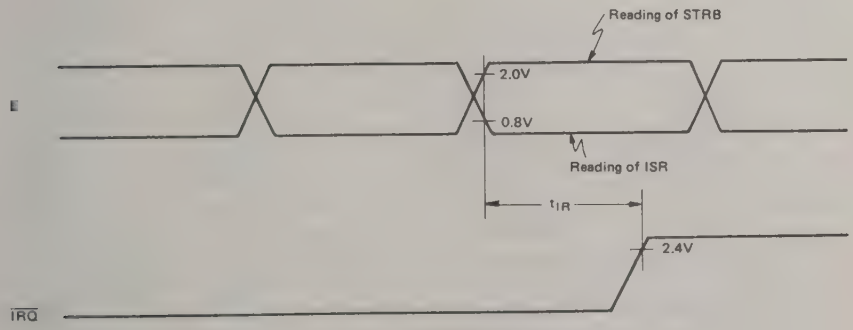


Figure 6 $\overline{IRQ}$ Release Timing

Figure 7 WDT Waveform

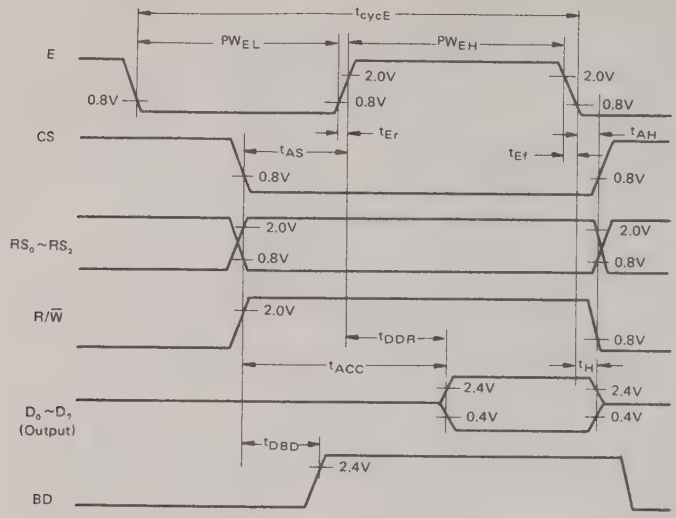


Figure 8 Read Timing

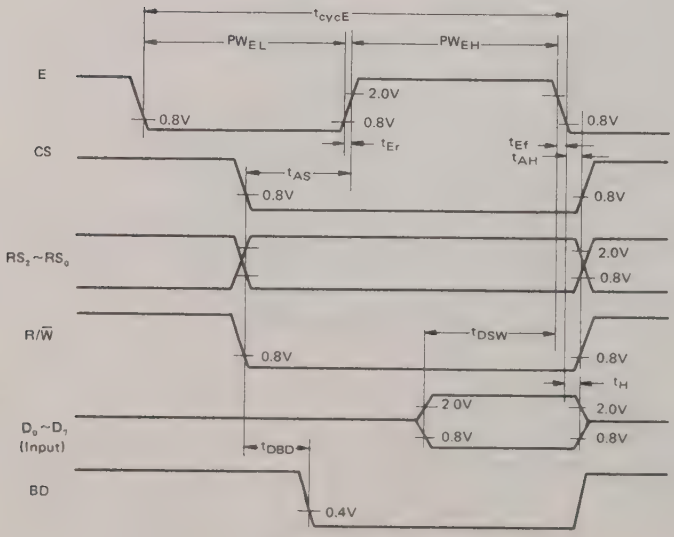


Figure 9 Write Timing

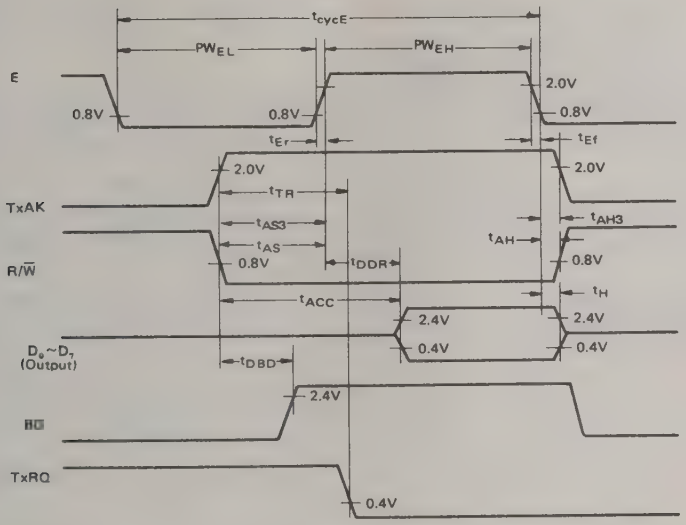


Figure 10 DMA Read Timing

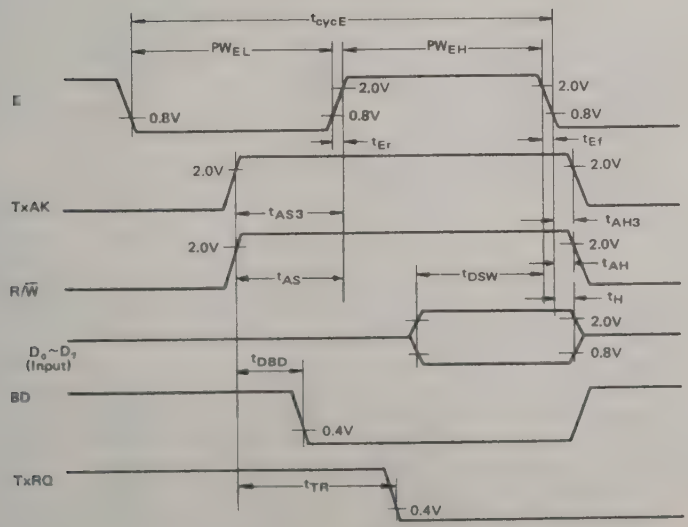


Figure 11 DMA Write Timing

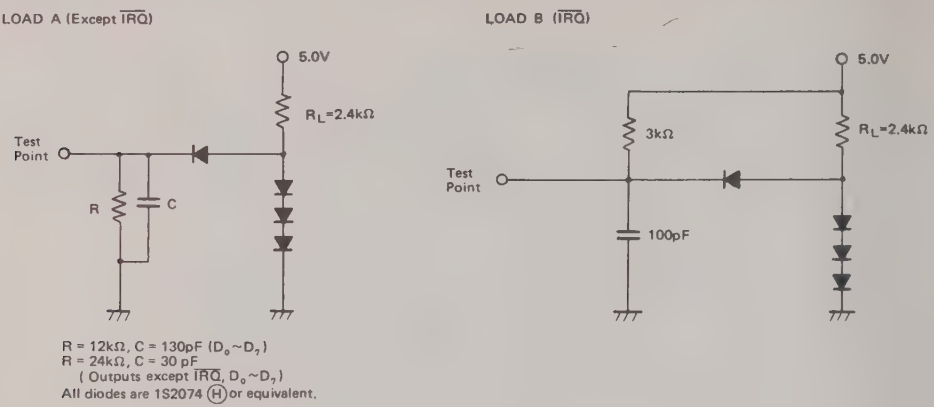


Figure 12 Load Circuit

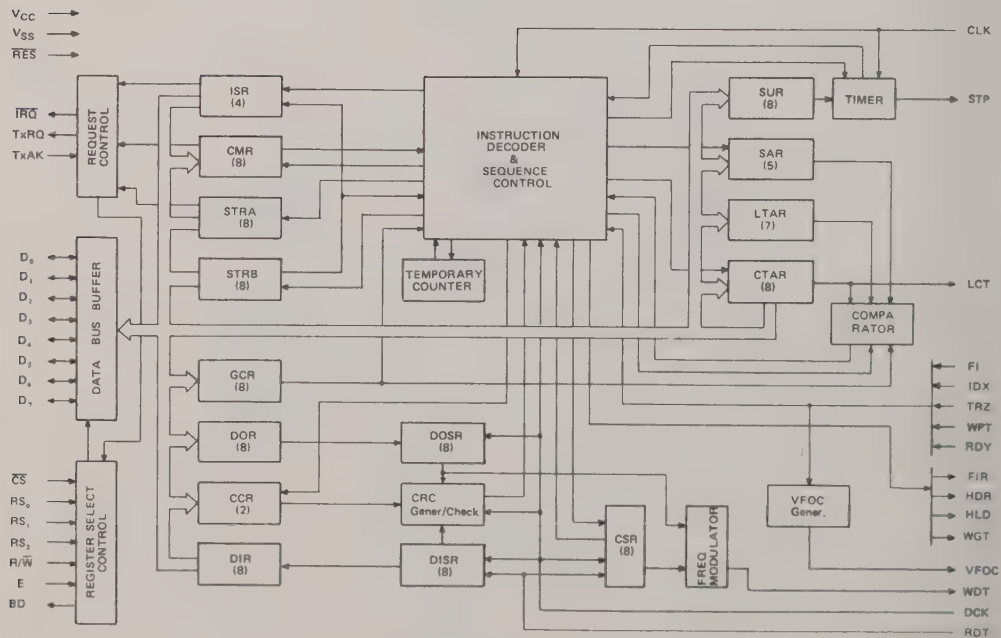


Figure 13 Block Diagram of the FDC

■ GENERAL DESCRIPTION

The HD6843 FDC consists of four primary sections; the Register, Serializing, Bus Interface, and Control sections. The following explanation of these sections can be followed in the block diagram of Figure 13.

• Register Section

The register section consists of twelve user accessible registers used for controlling a floppy disk drive. All twelve are connected by the internal data bus to allow the processor access to them.

Data Output Register (DOR)

The DOR is an 8-bit register which holds the data to be written onto the disk. The information is stored here by the bus interface.

Data Input Register (DIR)

The data words read from the disk are stored in the 8-bit DIR until read by the bus interface.

Current Track Address Register (CTAR)

CTAR is a 8-bit register containing the address of the track over which the R/W head is currently positioned.

Command Register (CMR)

The macro commands are written to the 8-bit CMR to begin their execution.

Interrupt Status Register (ISR)

The four bits of the ISR represent the four conditions that can cause an interrupt to occur.

Set-Up Register (SUR)

Variable Seek and Settling times are programmed by the SUR. Four bits are used to program the track to track seek time and four bits are used to program the head settling time for the floppy disk drive used with the FDC.

Status Register A (STRA)

The eight bits of STRA are used to indicate the state of the floppy disk interface.

Sector Address Register (SAR)

SAR contains the five bit sector address associated with the current data transfer.

Status Register B (STRB)

The eight error flags of STRB are used to signify error conditions detected by the FDC or generated by the floppy disk drive.

General Count Register (GCR)

The seven bits of GCR contain the destination track address when a SEK (seek) macro command is being executed. If a multi-sector Read or Write macro command is being executed, GCR contains the number of sectors to be read or written.

CRC Control Register (CCR)

The two bits of the CCR are used to enable the CRC and shift the CRC for the Free Format Commands.

Logical Track Address Register (LTAR)

The seven bit track address used for read and write operations is stored in the LTAR by the bus interface.

• Serializing Section

The serializing section handles the serial-to-parallel and parallel-to-serial conversions for Read/Write operations as well as CRC generation/checking and the generation/detection of the clock pattern. The Data Output Shift Register (DOSR), Data Input Shift Register (DISR), CRC Generator/Checker, and Clock Shift Register (CSR) comprise the serializing section of the FDC.

• Bus Interface

The Bus Interface section provides the timing and control logic that allows the FDC to operate with the 6800 bus, and is comprised of the Data Buffers. Request Control, and the Register Select circuitry.

• Control

The internal timing and control signals which sequence the FDC are derived from the macro instructions by the control section.

■ HD6843 PIN DESCRIPTION

• Power Pins

V_{CC}: +5 volt (±5%) power input.
V_{SS}: Power Supply Ground.

• Bus Pins

Reset ($\overline{RES}$) Input

The $\overline{RES}$ input is used to initialize the FDC. When $\overline{RES}$ becomes "Low", the state of the outputs is defined by the table below:

Output	State of Output	Output	State of Output
FIR	"Low"	HLD	"Low"
WGT	"Low"	TxRQ	"Low"
HDR	"Low"	$\overline{IRQ}$	"High"
STP	"Low"	WDT	"Low"

Registers which are affected by $\overline{RES}$ are shown in Table 7.

Interrupt Request ($\overline{IRQ}$) Output

The $\overline{IRQ}$ line is an open drain output that becomes a "Low" level (logic "0") when the FDC requests an interrupt. Interrupt requests are controlled by the interrupt enables in CMR (Command Register) with the function causing the interrupt shown in ISR (Interrupt Status Register).

Data Bus 0~Data Bus 7 (D_0~D_7) Bidirectional

The 8 bidirectional data lines allow the transfer of data between the FDC and the controlling system. The output buffers are three-state drivers that are enabled when the FDC is transferring data to the data bus.

Enable (E) Input

The E input to the FDC causes data transfers to occur between the FDC and the system controlling the FDC

(HMCS6800 MPU, DMA Controller, etc.) E must be a logic "1" ("High" level) for any transfer to be enabled on $D_0 \sim D_7$. The E input is normally connected to system ϕ_2.

Chip Select ($\overline{CS}$) Input

The $\overline{CS}$ input in conjunction with the E input, is used to enable data transfers on $D_0 \sim D_7$. E must be a "High" level and $\overline{CS}$ must be a "Low" level (logic "0") to enable the transfer. The TxAK input being a "High" level (logic "1") performs a similar function as $\overline{CS}$ being a "Low" level.

Read/Write (R/$\overline{W}$) Input

The R/$\overline{W}$ input is issued by the system controlling the FDC (HMCS6800 MPU, DMA Controller, etc.) to signify if a read or write operation is to be performed on the FDC. When TxAK is a "Low" level, R/$\overline{W}$ is used in conjunction with $\overline{CS}$ and $RS_0 \sim RS_2$ to determine which register is accessed by the bus as shown in Table 1. When TxAK is a "High" level, R/$\overline{W}$ is used to select either the DOR or DIR to the data bus (see description of TxAK input).

Register Select 0~Register Select 2 ($RS_0 \sim RS_2$) Input

$RS_0 \sim RS_2$, in conjunction with the R/$\overline{W}$ input, are used to select one of the user accessible registers in the FDC as shown in Table 1.

Transfer Request (TxRQ) Output

TxRQ is used in the DMA mode to request a data transfer from the DMAC. TxRQ is a "High" level if the FDC is in the DMA mode (CMR bit 5 is set) when a data transfer request occurs (STRA bit 0 is set). It is reset to a "Low" level (logic "0") when TxAK becomes a "High" level (logic "1"). Data transfer errors will occur if TxAK does not reset TxRQ before the next data transfer is required.

Transfer Acknowledge (TxAK) Input

TxAK is generated by the system controlling the FDC (HMCS6800 MPU, DMA Controller, etc.) and is a response to a TxRQ issued by the FDC. A "High" level (logic "1") on TxAK causes the FDC to neglect the state of $RS_0 \sim RS_2$ causing the FDC to select the DOR (Data Output Register) or DIR (Data Input Register) to the data bus ($D_0 \sim D_7$) as shown in Table 2. $\overline{CS}$ = "0" and TxAK = "1" cannot be permitted at the same time.

Table 2 Register Selection for DMA Transfers

TxAK	$RS_0 \sim RS_2$	$\overline{CS}$	R/$\overline{W}$	Register Selected
1	X	1	1	DOR
1	X	1	0	DIR

"1" ..... "High", "0" ..... "Low"

This mode of operation is normally used for DMA (Direct Memory Access) transfer with the FDC.

When TxAK is a "Low" level the registers are selected by $\overline{CS}$, R/$\overline{W}$ and $RS_0 \sim RS_2$ as shown in Table 1.

- **Bus Direction (BD) Output**

The BD output is provided to control external bidirectional buffers on the data bus ($D_0 \sim D_7$) as shown in Figure 14. Its polarity is shown by Table 3.

Table 3 Bus Direction (BD) States

TxAK	$\overline{CS}$	BD
1	1	R/$\overline{W}$
0	1	0
0	0	R/$\overline{W}$

"1" ..... "High", "0" ..... "Low"

(Operation of BD as defined by this chart allows the FDC to function with the DMA Controller HD6844.)

Table 1 Address Codes for User Accessible Registers

TxAK	$\overline{CS}$	RS_2	RS_1	RS_0	R/$\overline{W}$	Registers
0	0	0	0	0	0	DOR (Data Output Register)
					1	DIR (Data Input Register)
0	0	0	0	1	1/0	CTAR (Current Track Address Register)
0	0	0	1	0	0	CMR (Command Register)
					1	ISR (Interrupt Status Register)
0	0	0	1	1	0	SUR (Set Up Register)
					1	STRA (Status Regiser A)
0	0	1	0	0	0	SAR (Sector Address Register)
					1	STRB (Status Register B)
0	0	1	0	1	0	GCR (General Count Register)
0	0	1	1	0	0	CCR (CRC Control Register)
0	0	1	1	1	0	LTAR (Logical Track Address Register)

"1" ..... "High", "0" ..... "Low"

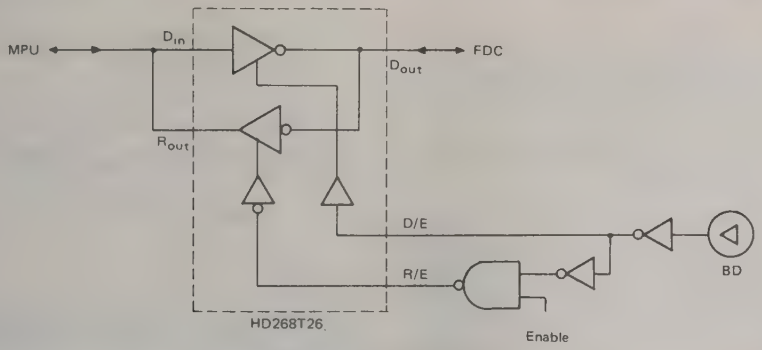


Figure 14 Bus Buffer Control

- **I/O and Control Pins**

Head Load (HLD) Output

HLD is used to notify the disk drive that the R/W head should be loaded (placed in contact with the media). When the FDC is ready for the head to load, HLD is a "High" level (logic "1"). A "Low" level (logic "0") HLD indicates the head should be unloaded.

Step (STP) Output

The STP output, in conjunction with HDR, is used to control head movement. A 32 μs wide positive (logic "1") pulse is generated on STP, to move the R/W head one track in the direction defined by the HDR output. The period of the STP signal is programmable by the SUR (Set-Up Register). The number of pulses generated on STP is the difference between the contents of the CTAR (Current Track Address Register), and the GCR (General Count Register) which contains the track address to which the head is to be moved.

Head Direction (HDR) Output

The HDR signal controls the direction of head movement. A "High" level (logic "1") signifies the head should step to the inside (toward the hub) of the disk. A "Low" level (logic "0") indicates the direction of head movement should be to the outside of the disk.

Low Current Track (LCT) Output

The LCT signal is used to control the level of write current used by the disk drive. LCT is a "Low" level (logic "0") when the write head is positioned over tracks 0~43. If it is over tracks 44~76, LCT is a "High" level (logic "1"). LCT is determined from the contents of the Current Track Address Register (CTAR).

Write Gate (WGT) Output

When a write operation is being performed, WGT is a logic "1" ("High" level). For a read operation, WGT is a "Low" level (logic "0").

File Inoperable Reset (FIR) Output

FIR is an output from the FDC to the floppy disk drive to reset it from an inoperable status. If the FI input is a "High" level, a pulse, of which width almost equals to E pulse "Low" width, is generated on the FIR output whenever Status Register B is read.

File Inoperable (FI) Input

FI is an input to the FDC from the drive. A "High" level indicates the drive is in an inoperable state. Its current state can be examined by reading bit 5 of Status Register B (STRB).

Track Zero (TRZ) Input

The TRZ input is reflected by bit 3 of STRA (Status Register A). The TRZ input must be a "High" level (logic "1") when the R/W head of the drive is positioned over track zero. A logic "1" on this input inhibits step pulses during a Seek Track Zero command.

Index (IDX) Input

The index input is received from the floppy disk drive and is used to sense the index hole in the disk media. The IDX signal is used to initialize the internal FDC timing. The state of the IDX input is reflected by bit 6 of Status Register A (STRA). A "High" level (logic "1") is to indicate the index hole is under the index sensor. The index input is used to count the number of disk revolutions while searching for the address ID field (see description of STRB bit 3).

Ready (RDY) Input

The ready input is received from the disk drive and can be read as bit 2 of STRA (Status Register A). A "High" level (logic "1") indicates the drive is ready and allows the FDC to operate the drive.

Write Protect (WPT) Input

WPT is an input indicating when the media is Write Protected. A "High" level during an FDC write operation results in a Write Error (STRB bit 6) but the FDC continues to perform the write function. The state of the WPT input can be read by examining bit 4 of the Status Register A (STRA).

Clock (CLK) Input

The CLK input is used to generate various timing sequences internal to the FDC. The head settling, seek time, step pulse width and write data pulse width, etc., are generated from the CLK input signal. The CLK is 1 MHz frequency and the duty is 50%.

- **Data Pins**

Data Clock (DCK) Input

DCK is used to clock data from the drive into the FDC. It is generated from the read data received from the drive.

Read Data (RDT) Input

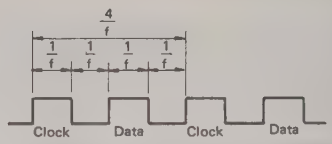
RDT is the serial data input from the drive. The data stream includes both the clock and data bits.

Write Data (WDT) Output

WDT is the double frequency modulated data output from the FDC. The time between clock bits is 4/f where f is the frequency of the clock input. The pulse width for both clock and data is 1/f (see Figure 15). For the normal clock frequency of 1 MHz the clock period is 4 μs, the clock pulse width is 1 μs and the data pulse width is 1 μs. Figure 15 shows the relationship between the WDT output and the frequency of the CLK inputs.

Variable Frequency Oscillator Control (VFOC) Output

VFOC is used as a sync signal during system diagnostics. Waveforms are shown in Figure 16.



f = Frequency of the CLK Input. To insure IBM3740 compatibility the clock frequency must be 1 MHz.

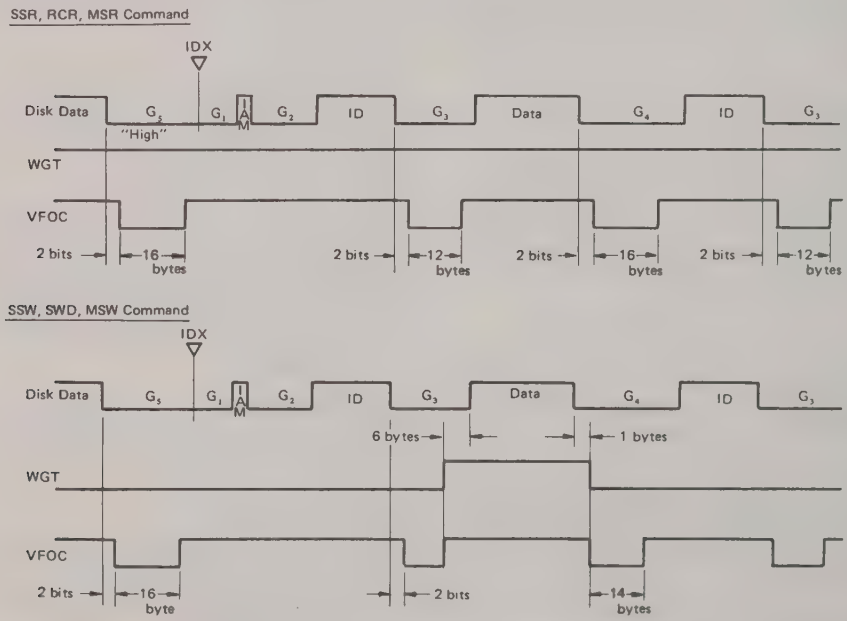
Figure 15 WDT Output Timing

■ FORMAT

The format used by the HD6843, shown in Figure 18, is compatible with the soft sector format of the IBM3740.

■ MACRO COMMAND SET

The macro command set shown in Table 4 is discussed in the following paragraphs.



In FFW Command, VFOC becomes "High" when WGT is at "High" level.
In FFR Command, VFOC remains "High".

Figure 16 Variable Frequency Oscillator Control Waveform (Relation Between WGT and VFOC)

SSW, SWD and MSW commands (Single Sector Write, Single Sector Write with Delet Data Mark, and Multi-Sector Write)

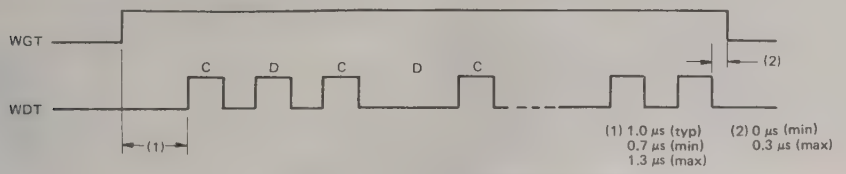


Figure 17 Write Data versus Write Gate Timing

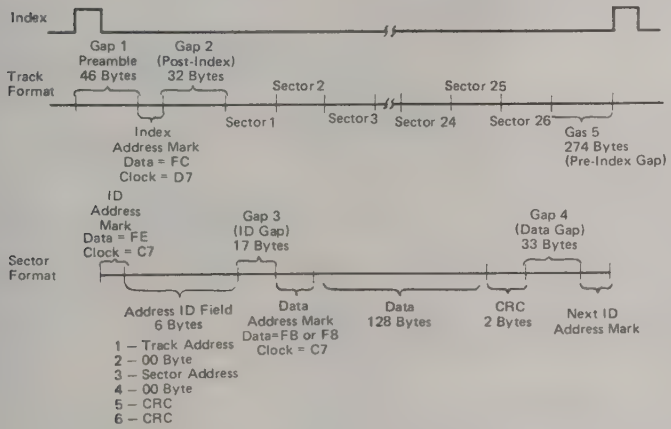

Index
Track Format
Gap 1 Preamble 46 Bytes
Gap 2 (Post-Index) 32 Bytes
Sector 1
Sector 2
Sector 3
Sector 24
Sector 25
Sector 26
Gas 5 274 Bytes (Pre-Index Gap)
Index Address Mark Data = FC Clock = D7
Sector Format
ID Address Mark Data = FE Clock = C7
Gap 3 (ID Gap) 17 Bytes
Gap 4 (Data Gap) 33 Bytes
Address ID Field 6 Bytes
1 — Track Address
2 — 00 Byte
3 — Sector Address
4 — 00 Byte
5 — CRC
6 — CRC
Data Address Mark Data=FB or F8 Clock = C7
Data 128 Bytes
CRC 2 Bytes
Next ID Address Mark

Figure 18 Soft Sector Format

Table 4 Macro Command Set

	Macro Command		CMR Bits				Hex Code
			Bit 3	Bit 2	Bit 1	Bit 0	
1	STZ	Seek Track Zero	0	0	1	0	2
2	SEK	Seek	0	0	1	1	3
3	SSR	Single Sector Read	0	1	0	0	4
4	SSW	Single Sector Write	0	1	0	1	5
5	RCR	Read CRC	0	1	1	0	6
6	SWD	Single Sector Write with Delete Data Mark	0	1	1	1	7
7	MSW	Multi Sector Write	1	1	0	1	D
8	MSR	Multi Sector Read	1	1	0	0	C
9	FFW	Free Format Write	1	0	1	1	B
10	FFR	Free Format Read	1	0	1	0	A

- **Seek Track Zero (STZ)**

The STZ command causes the R/W head to be released from the surface of the disk (HLD is reset) and positioned above track 00. The FDC issues step pulses on the STP output until the TRZ input becomes a "High" level or until 82 pulses have been sent to the drive. When the TRZ input becomes "High", the step pulses are inhibited on the STP output but the FDC remains busy until all 82 have been generated internally.

If the TRZ input remains "Low" (logic "0") after all 82 pulses have been generated, the seek error flag (STRB bit 4) is set.

After all 82 pulses have been generated, the head is loaded (HLD becomes a "High"). After the settling time specified in the SUR has expired, the Seek Command End flag is set (ISR bit 1), Busy STRA7 is reset, CTAR and GCR are cleared. The head remains in contact with the disk. A command such as RCR (Read CRC) may be issued following a STZ if the head must be released.

- **Seek (SEK)**

The SEK command is used to position the R/W head over the track on which a Read/Write operation is to be performed. The contents of the GCR are taken as the destination address and the content of the CTAR is the source address; therefore, the number of pulses (N) on the STP output are given by:

$$N = |(CTAR) - (GCR)|$$

HDR is a "High" for (GCR) > (CTAR) otherwise it is a "Low".

When a SEK command is issued, Busy is set, the head is raised from the disk, HDR is set as described above, and N number of pulses appear on the STP output. After the last step pulse is used, the head is placed in contact with the disk. Once the head settling time has expired, the Seek Command End flag (ISR bit 1) is set, Busy is reset, and the contents of the GCR are transferred to the CTAR.

■ SINGLE SECTOR READ/WRITE COMMANDS

The single sector Read/Write commands (SSR, RCR, SSW, and SWD) are used to Read/Write data from a single 128 byte sector on the disk. As shown in Figure 19 these types of instructions can be divided into two sections. The first section, which is common to all instructions, is the address search operation, while the second section is unique to the requirements of each instruction.

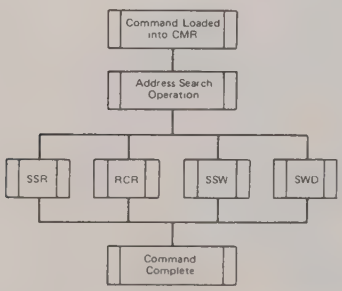


Figure 19 Basic Single Sector Command Flow Chart

- **Address Search Operation**

The flow chart of Figure 20 shows the operation of the address search operation.

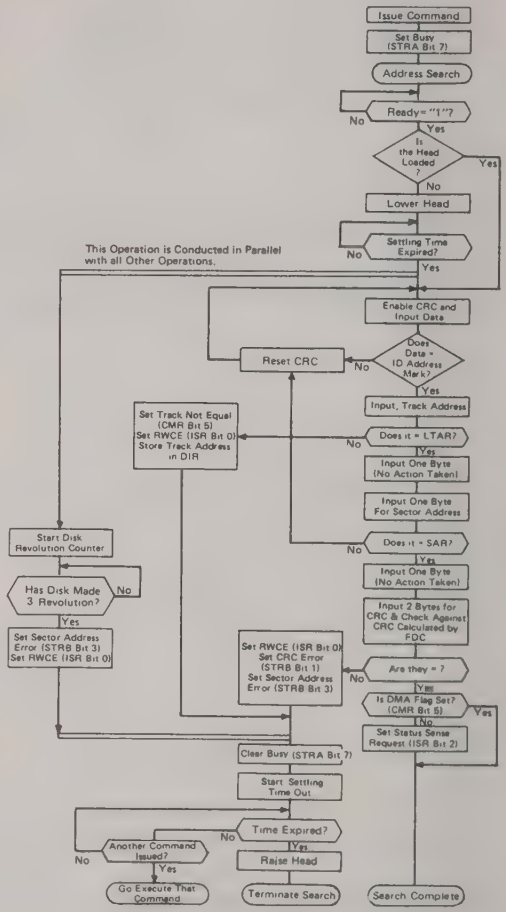


Figure 20 Operational Flow of the Address Search Sequence

- **Single Sector Read (SSR)**

The single sector read command follows the address search procedure as defined in the previous flowchart. If the search is successful, status sense request is set and the operation continues as described by the flowchart of Figure 21 .

- **Read CRC (RCR)**

The RCR command is used to verify that correct data was written on a disk. The operation is the same as for the SSR

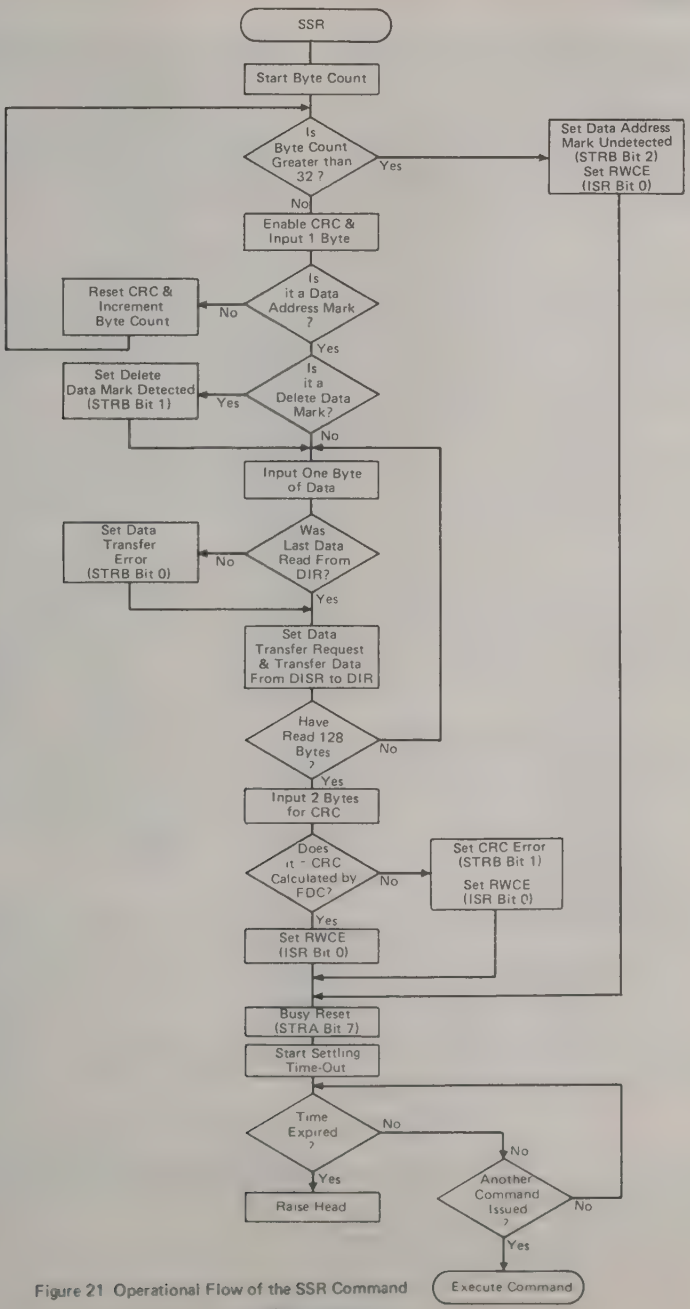


Figure 21 Operational Flow of the SSR Command

command with the exception that the data transfer request (STRA bit 0) is not set. The Status Sense Request interrupt can be disabled by using the DMA flag of CMR.

- **Single Sector Write (SSW)**

Single sector write is used to write 128 bytes of data on the disk. After the command is issued, the address search is performed. The remainder of the instruction's operation is shown in Figure 22.

- **Single Sector Write with Delete Data Mark (SWD)**

The operation flow of SWD is exactly like that of SSW. For SWD, the data pattern of the Data Address Mark becomes F8 instead of FB. The clock pattern remains C7.

- **Multi-Sector Commands (MSR/MSW)**

MSR is used for sequential reading of one or more sectors. If S sectors are to be read, S – 1 must be written into the GCR before the command is issued.

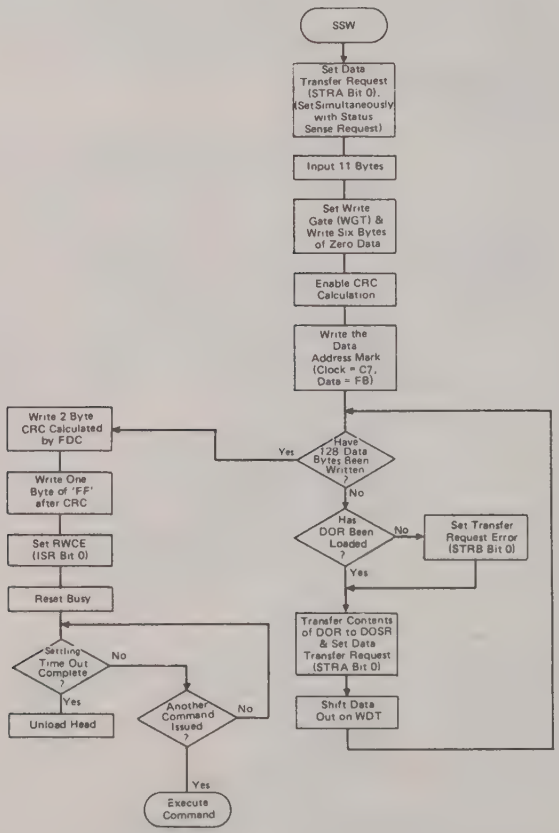

SSW
Set Data Transfer Request (STRA Bit 0). (Set Simultaneously with Status Sense Request)
Input 11 Bytes
Set Write Gate (WGT) & Write Six Bytes of Zero Data
Enable CRC Calculation
Write the Data Address Mark (Clock = C7, Data = FB)
Write 2 Byte CRC Calculated by FDC
Yes
Have 128 Data Bytes Been Written ?
No
Write One Byte of 'FF' after CRC
Has DOR Been Loaded ?
No
Set Transfer Request Error (STRB Bit 0)
Yes
Set RWCE (ISR Bit 0)
Reset Busy
Transfer Contents of DOR to DOSR & Set Data Transfer Request (STRA Bit 0)
Seeking Time Out Complete ?
No
Yes
Another Command Issued ?
No
Yes
Shift Data Out on WDT
Unload Head
Execute Command

Figure 22 Operational Flow of the SSW Command

The basic operation for the MSR and MSW is the same as that for the SSR and SSW respectively. The basic operation begins with an address search operation, which is followed by a single sector read or write operation. This completes the operation on the first sector. The SAR is incremented, the GCR is decremented, and if no overflow is detected from the GCR (i.e., GCR become negative) the sequence is repeated until S number of sectors are read or written.

The completion of an MSR or MSW is like that of an SSR or SSW command. First RWCE is set and Busy is reset, after the settling time has expired, the head is released.

If a delete data mark is detected during an MSR command, STRA bit 1 (Delete Data Mark Detected) remains set throughout the commands operation.

When a multi-sector instruction is issued, the sum of the SAR and GCR must be less than 27. If SAR + GCR > 26, an address error (STRB bit 3 set) will occur after the contents of SAR becomes greater than 26.

- **Free Format Write (FFW)**

The FFW has two modes of operation which are selected by FWF (Free Format Write Flag) which is data bit 4 of the CMR.

When FWF = "0", the data bits of the DOR are written directly to the disk without first writing the preamble, address mark, etc. The contents of the DOR are FM modulated with a clock pattern of all ones.

If FWF = "1" the odd bits of the DOR are used as clock bits and even bits are used for data bits. In this mode, the DOSR clock is twice a normal write operation and one byte of DOR is one nibble (four bits of data) on the disk.

The two modes of the FFW command allow formatting a disk with either the IBM3470 format or a user defined format.

After the FFW command is loaded into the CMR, WGT becomes a "High" level, the contents of DOR are transferred to the DOSR, data transfer request (STRA bit 0) is set, and the serial bit pattern is shifted out on the WDT line. Therefore, DOR must be loaded before the FFW command is issued. Data from the DOR is continually transferred to the DOSR and shifted out on WDT until the CMR has been written with an all zero pattern. When CMR becomes zero, WGT becomes a "Low" level, but RWCE is not set and the R/W head is left in contact with the disk.

- **Free Format Read (FFR)**

FFR is used to input all data (including Address marks) from a disk. Once the FFR command is set into the CMR, the head is loaded and after the settling time has expired the serial data from the FDC is brought into the DISR. After 8 bits have accumulated, it is transferred to the DIR and Data Transfer Request (STRA bit 0) is set.

This operation continues until a zero pattern is stored in the CMR, terminating the FFR command. As in the case of the FFW command, RWCE is not set and the head remains in contact with the disk.

The first data that enters the DISR is not necessarily the first bit of a data word since the head may be lowered at any place on the disk. To prevent the FDC from remaining unsynchronized to the data, the FFR command will synchronize to an ID address mark (FE) or a Data Address mark (FB or F8) or an Index Address Mark (FC).

■ REGISTER DEFINITIONS

- **Data Output Register (DOR); Hex address 0, write only**

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
8 Bits of Data Used for a Disk Write Operation							

When one of the four write macro commands (SSW, SWD, MSW, and FFW) is executed, the information contained in the DOR is loaded into the DOSR, and is shifted out on the WDT line using a double frequency (FM) format.

- **Data Input Register (DIR); Hex address 0, read only**

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
8 Bits of Data Used for a Disk Read Operation							

One of the three read macro commands (SSR, MSR, FFR) executed, will cause the information on the RDT input to be clocked into the DISR. When 8 clock pulses have occurred, the 8 bits of information in the DISR are transferred to the DIR where it can be read by the bus interface.

- **Current Track Address (CTAR); Hex address 1, read/write**

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Track Address of Current Head position							

The address of the track over which the R/W head is currently positioned is contained in the CTAR. At the end of a SEK command, the contents of the GCR are transferred to the CTAR. CTAR is cleared at the completion of a STZ command. CTAR is a read/write register so that the head position can be updated when several drives are connected to one FDC. Bit 7 is read as a "0".

- **Command Register (CMR); Hex address 2, write only**

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3*	Bit 2*	Bit 1*	Bit 0*
Function Interrupt Mask	ISR3 Interrupt Mask	DMA Flag	FWF	Macro Command			

*Bit 0 ~ 3 are cleared by $\overline{RES}$.

The commands that control the FDC are loaded into the lower four bits of the CMR. Information that controls the data transfer mode and interrupt conditions are loaded into bits four through seven.

Bit 0~Bit 3: Macro Command

The Macro Command to be executed by the FDC is written to bits 0~3.

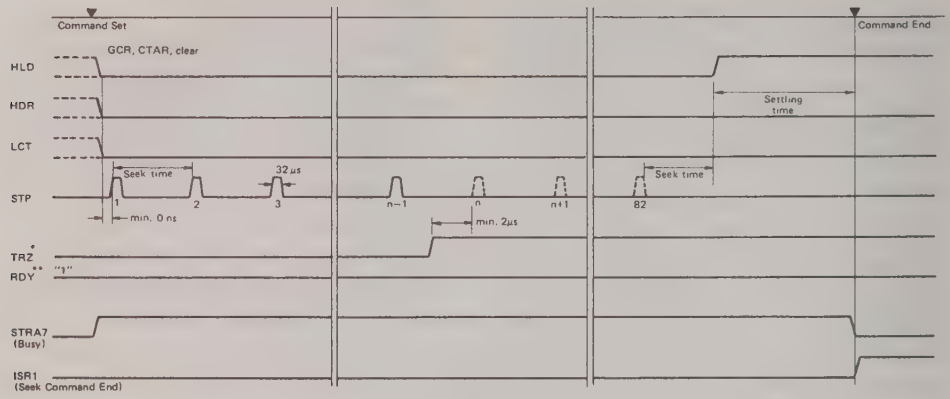
Bit 4: Free Format Write Flag (FWF)

If a Free Format Write command is issued, the state of bit 4 of the CMR determines what clock source will be used. The FWF is defined in the FFW (Free Format Write) command explanation.

Bit 5: DMA Flag

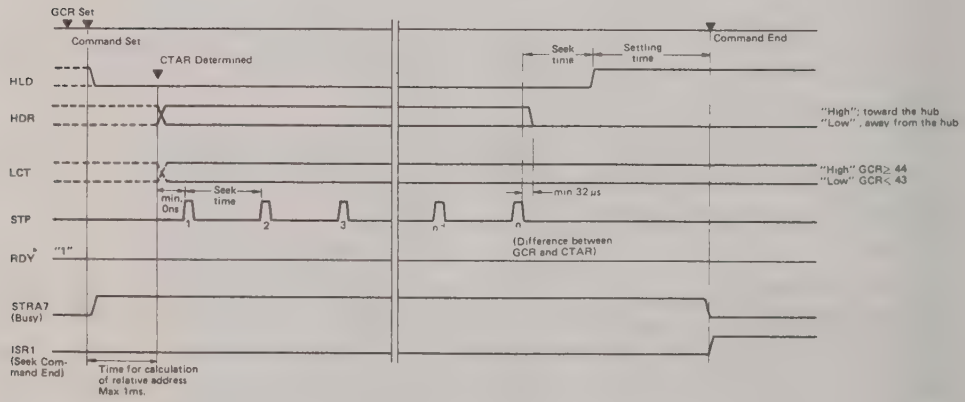
If bit 5 is a "1" the FDC is in the DMA mode. Bit 5 being a "1" inhibits setting of Status Sense Request (ISR bit 2) thereby preventing its associated interrupt. A logic "1" DMA flag also enables the TxRQ output allowing it to request DMA transfers when the Data Transfer Request flag (STRA bit 0) is set.

A logic "0" DMA flag indicates the program controlled I/O (PC I/O) mode.



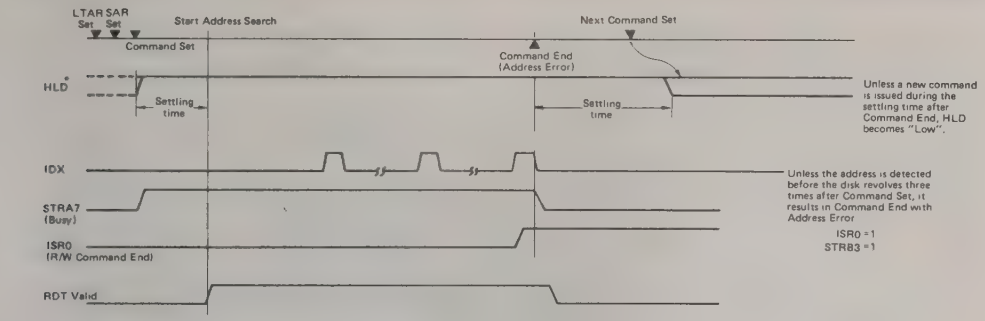
* STP output is masked when TRZ becomes "High". But if TRZ falls to "Low" again before 82 pulse outputs are all provided, STP output become available again from that time point.
** When RDY is "Low" with Command Set, the execution is postponed until RDY becomes "High".

Figure 23 Timing Sequence of STZ Command



* When RDY is "Low" with Command Set, the execution is postponed until RDY becomes "High".

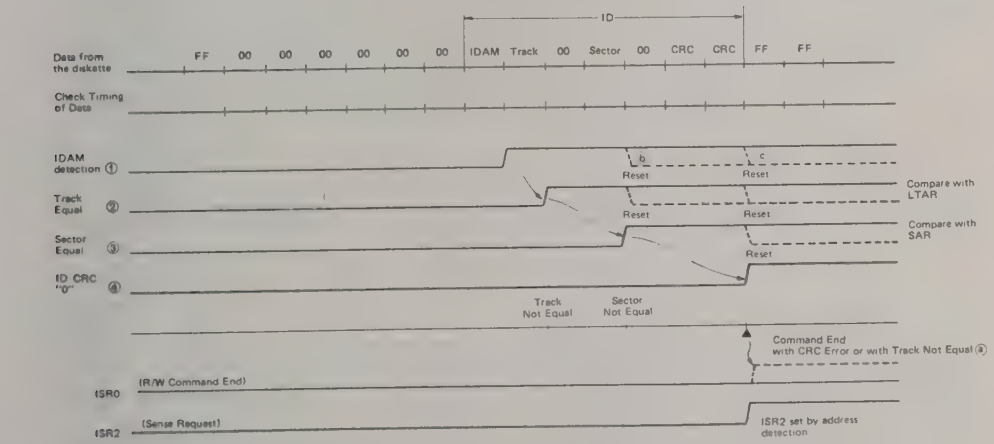
Figure 24 Timing Sequence of SEK Command



* If HLD has already been "High" when the command is set, the FDC starts the address search immediately.

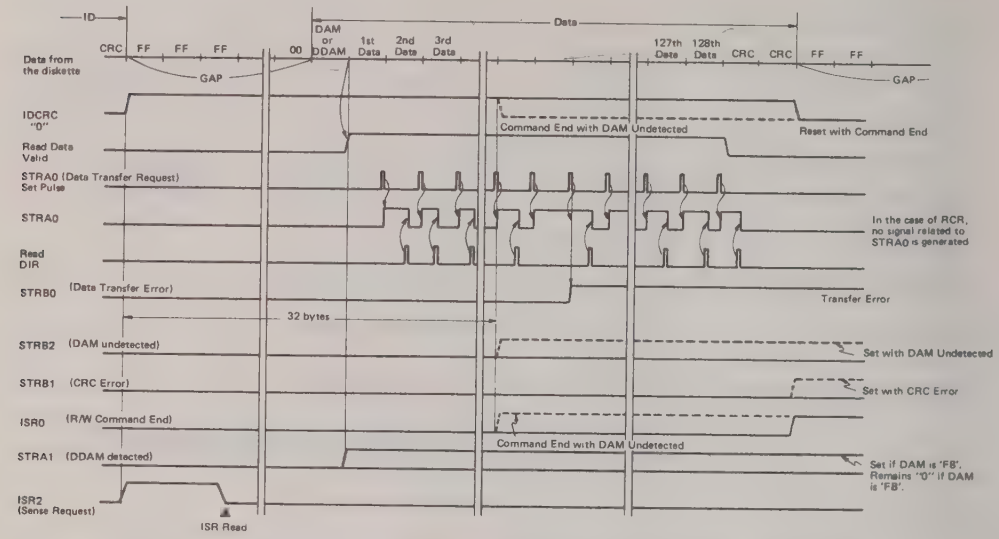
When RDY is "Low" with Command Set, the FDC waits for the execution until RDY becomes "High".

Figure 25 Timing Sequence of SSR, SSW, RCR, SWD, MSR, MSW Command (Relation with HLD and IDX)



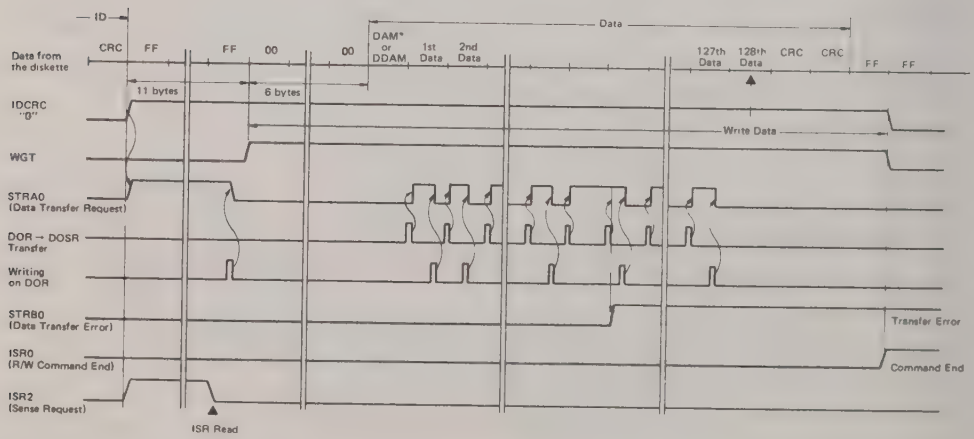
ⓐ ; In the case of Track Not Equal, ② is not set and if CRC equals to the one calculated by FDC, STRA5 is set.
ⓑ , In the case of Sector Not Equal, ③ is not set and ① & ② are reset to search the next IDAM.
ⓒ , In the case of CRC Error, ④ is not set and ① , ② & ③ are reset. (ISR0 Set, STRB1 Set, STRB3 Set)
When ① , ② , ③ , & ④ are all set, ISR2 is Set. These four signals are reset with Command End.
When ④ is "1", go to the data transfer routine.

Figure 26 Internal Timing Sequence of Address Search Routine



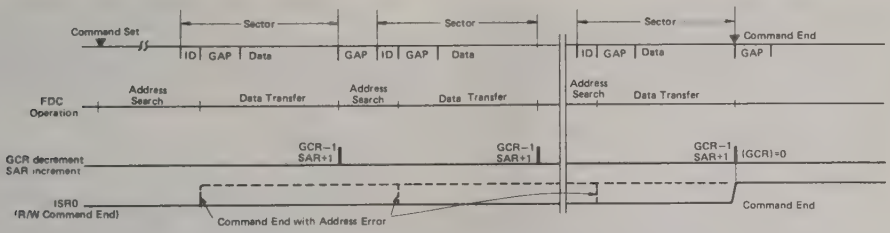
Unless DAM(F8) or DDAM(F8) is detected within 32 bytes after ID field has been detected, STRB2 is set to end the command.

Figure 27 Data Transfer Timing of SSR, RCR Command



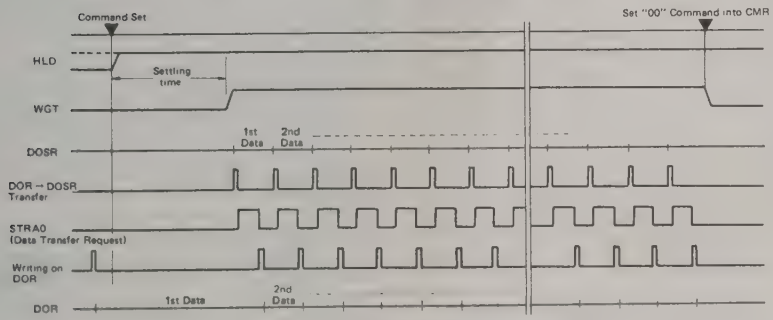
* As Data Address Mark, SSW command writes 'F8' and SWD command writes 'F8'.

Figure 28 Data Transfer Timing of SSW, SWD Command



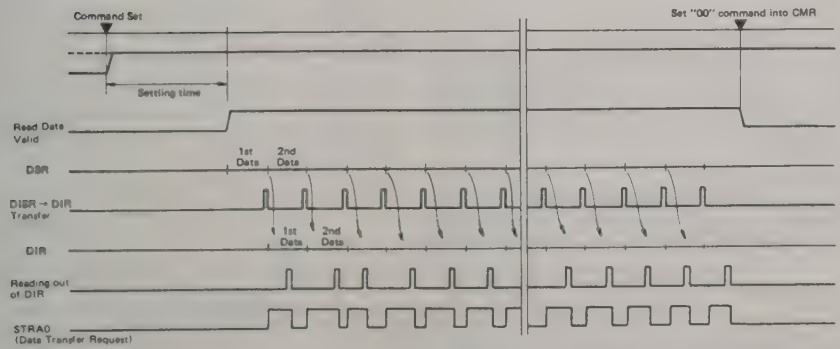
Address Search and Data Transfer in each sector is the same as those of SSR or SSW command. When Address Error occurs, it results in Command End. If an error relating to Data Transfer occurs, Error flag is set. But the command continues to be executed to shift into the next sector.

Figure 29 Timing Sequence of MSR, MSW Command



- The first one-byte data must be set into DOR before Command Set.
- If HLD has already been "High" when the command is set, WGT becomes "High" immediately.
- When '00' command is set into CMR, an interrupt of Command End is not generated.

Figure 30 Timing Sequence of FFW Command



If HLD has already been "High" when the command is set, Read operation starts immediately without waiting for the settling time. When "00" command is set into CMR, an interrupt of Command End is not generated.

Figure 31 Timing Sequence of FFR Command

Bit 6: ISR3 Interrupt Mask

CMR bit 6 (ISR3 Mask) is used to control the operation of ISR bit 3. A logic "1" in CMR bit 6 inhibits output of STRB-OR-Interrupt signal to $\overline{IRQ}$. If CMR bit 6 (ISR3 Mask) and CMR bit 7 are "0" STRB-OR-Interrupt signal will be output to $\overline{IRQ}$.

Bit 7: Function Interrupt Mask

When CMR bit 7 is a logic "1" all interrupts are inhibited.

Table 5

Causes of Interrupt	Command Register Masks That Affect Interrupts		
	CMR7 (Function Interrupt Mask)	CMR6 (ISR3 Mask)	CMR5 (DMA Flag)
ISR0 (Read write Command End)	M	X	X
ISR1 (Seek Command End)	M	X	X
ISR2 (Status Sense Request)	M	X	M
ISR3 (STRB-OR-Interrupt)	M	M	X

X = No effect
M = Bits that are used as masks

- **Interrupt Status Register (ISR); Hex address 2, read only**

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2*	Bit 1*	Bit 0*
Not Used (Read as "0")				STRB -OR	Status Sense Request	Seek Command End	Read Write Command End

* Cleared by $\overline{RES}$

Bit 0: Read Write Command End (RWCE)

When an SSR, RCR, SSW, SWD, MSR or MSW Macro Command has completed execution, bit 0 becomes set (logic "1"). If the function interrupts are enabled (bit 7 of CMR is a logic "0"), the conclusion of a Macro Command's execution will cause an interrupt.

Bit 1: Seek Command End (SCE)

Seek Command End is set on SEK and STZ commands to indicate the head has been loaded and the settling time specified in SUR has expired. Since RWCE is not set for the SEK or STZ command, SCE can be used as an interrupt to signify the SEK or STZ command has finished. SCE is not set for any of the R/W commands.

Bit 2: Status Sense Request

For an SSR, SSW, SWD, MSR, or MSW Command, Status Sense Request indicates that the specified address ID field has been detected and verified by a CRC check. This is used as an early indication that data transfers will occur after 18 more byte times. For MSR and MSW commands, it is set for each sector.

In the PC I/O mode, an interrupt occurs when Status Sense Request becomes a logic "1". In the DMA mode, (DMA flag of CMR is set) Status Sense Request is unchanged and does not generate an interrupt when the address ID field has been verified.

Bit 3: STRB-OR

STRB-OR is an "OR" of all of the bits of Status Register B.

$$\text{STRB-OR} = \text{STRB0} + \text{STRB1} + \text{STRB2} + \text{STRB3} + \text{STRB4} + \text{STRB5} + \text{STRB6} + \text{STRB7}$$

$$\text{STRB-OR-Interrupt} = \text{STRB1} + \text{STRB2} + \text{STRB3} + \text{STRB4} + \text{STRB5} + \text{STRB6} + \text{STRB7}$$

STRB-OR-Interrupt signal causes $\overline{IRQ}$. STRB-OR is read by Read ISR. STRB0 (Data Transfer Error) sets ISR Bit 3 but does not cause Interrupt.

ISR0, ISR1, and ISR2 are cleared when the Interrupt Status Register is read, but ISR3 is cleared only after Status Register B has been read except when FI input is "High".

- **Set-Up Register (SUR); Hex address 3, write only**

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Track to Track Seek Time				Head Settling Time			

The SUR is not affected by a reset operation; therefore, once it is initialized, the information remains until power is removed from the FDC.

Bit 0 ~ Bit 3: Head Settling Time

The head settling time is used to generate a delay after the head is placed in contact with the disk. This allows the head to stop bouncing before any operations are performed. The delay is programmed by bits 0~3 and is specified by the equation:

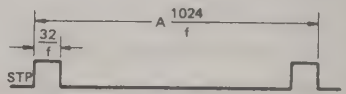
$$\text{Delay} = \frac{4096}{f} \cdot B$$

B = Number contained in bits 0~3 of SUR
f = Frequency of CLK input

For IBM3740 compatibility f = 1 MHz and the timing range is 4.096 ms for a "0001" to 61.44 ms for a "1111". A "0000" code prevents Settling Time complete from being set and the FDC must be Reset.

Bit 4 ~ Bit 7: Track to Track Seek Time

The frequency of STP is determined by bit 4~ bit 7 of SUR as shown below.



A = Number specified in bits 4~7 of SUR.
f = Frequency of CLK input.

For IBM compatible operation, f is 1 MHz. This results in an STP pulse width of 32 μs and an STP interval of 1.024 ms for a "0001" to 15.36 ms for a "1111".

- **Status Register A (STRA); Hex address 3, read only**

Bit 7*	Bit 6	Bit 5*	Bit 4	Bit 3	Bit 2	Bit 1*	Bit 0*
Busy	Index	Track Not Equal	Write Protect	Track Zero	Drive Ready	Delete Data Mark Detected	Data Transfer Request

* Cleared by $\overline{RES}$

Bit 0: Data Transfer Request

For a write operation (SSW, SWD, MSW, FFW) the transfer request bit indicates that the DOR is ready to accept the next data word to be written on the disk. If data is not written into the DOR before the last data bit in the DOSR is shifted out to the WDT line; the data transfer error bit (bit 0 of STRB) will be set. After a write command has been issued, the first transfer request occurs simultaneously with the Status Sense Request. For a write operation, transfer request is reset after the DOR has been written from the data bus.

During a read operation (SSR, MSR, FFR) the transfer request bit signifies data from the DISR has been transferred to the DIR. The DIR must be read before the DISR is full again or the data transfer error bit (bit 0 of STRB) will be set. For read operations, transfer request is reset by a read of the DIR.

Bit 1: Delete Data Mark Detected

A Single Sector Read operation that detects a delete data code (F8) instead of a general data code (FB) as a Data Address Mark will set the Delete Data Mark Detected bit. For the MSR command, bit 1 is set the first time an "F8" code is found and remains set throughout the execution of the command. Bit 1 is reset whenever an SSR, SSW, SWD, MSR, MSW, or RCR command is issued.

Bit 2: Drive Ready

The Drive Ready bit indicates the state of the Ready input from the floppy disk drive. If a command is issued with Ready at logic "0", its execution will be inhibited until Ready becomes a logic "1". If ready becomes a "0" during the execution of a command the Hard Error Flag (STRB bit 7) is set.

Bit 3: Track Zero

The state of the Track Zero input from the floppy disk drive is reflected in this bit of STRA. A logic "1" on the Track Zero input inhibits step pulses during an STZ command.

Bit 4: Write Protect

The Write Protect input from the floppy disk drive is reflected by bit 4 of STRA. A "High" level (logic "1") on the WPT input during the execution of any write command results in a write error (bit 6 of STRB set).

Bit 5: Track Not Equal

If the track address read from the address ID field does not coincide with the address in the LTAR inspite of CRC matching the one calculated by FDC, the Track Not Equal bit is set. Track Not Equal applies to all non-free format read/write commands, and is reset after a non-free format read/write command is issued.

Bit 6: Index

The state of the index input appears in bit 6 of STRA. The index input is used to count the number of disk revolutions while the FDC is looking for the address ID field (see operation of STRB bit 3) during the address search phase of a non-free format read/write command.

Bit 7: Busy

When Busy is a logic "1", the FDC is executing a command and no new commands can be issued. Busy should be confirmed to be "0" before reading ISR or STRB as well as issuing a command.

- **Sector Address Register (SAR); Hex address 4, write only**

Bit 7	Bit 6	Bit 5	Bit 4*	Bit 3*	Bit 2*	Bit 1*	Bit 0*
Not Used			5 Bit Sector Address				

* Cleared by $\overline{RES}$

Before a data transfer macro command (SSW, SWD, SSR, RCR, MSW, MSR) is issued, the address of the sector on which the operation is to be performed must be written into the SAR. The address in the sector address byte of an Address ID field of the disk is compared with the contents of the SAR. During an MSW or MSR command, the SAR is incremented after each sector is read or written. When execution is complete, the SAR contains the address of the last sector on which an operation was performed plus one.

- **Status Register B(STRB); Hex address 4, read only**

Bit 7*	Bit 6*	Bit 5	Bit 4*	Bit 3*	Bit 2*	Bit 1*	Bit 0*
Hard Error	Write Error	File Inoperable	Seek Error	Sector Address Undetected	Data Mark Undetected	CRC Error	Data Transfer Error

* Cleared by $\overline{RES}$

The bits of the STRB represent possible error conditions that may occur during execution of macro commands. Whenever STRB is reset, ISR bit 3 is also reset.

Bit 0: Data Transfer Error

Data Transfer Error indicates an underflow or overflow of data. If a Write operation is being performed, it signifies that data was not presented to the DOR before the DOSR became empty. In this case, the current contents of the DOR are transferred to the DOSR and the write operation continues. The data transfer error remains set until STRB is read, and the data transfer request remains set until data is written into the DOR. The operation of the CRC is unchanged.

For read commands, a data transfer error indicates that data in the DIR was not read before the next data word from the disk was transferred to the DIR. The read operation continues until sufficient data has been read from the disk to satisfy the requirements of the command (128 bytes for SSR). The error indication remains set until STRB is read, and the transfer request remains set until data is read from the DIR.

Bit 1: CRC Error

A CRC error occurs when the CRC read from the disk does not match that calculated by the FDC on the data it reads from the disk. A CRC error can occur in two different situations; checking the address ID field, checking the data field.

If the CRC error occurs during the check of an address ID field, Sector Address Undetected (STRB bit 3) will also be indicated (see Table 6). A CRC error of a data field is indicated by a CRC Error and no Sector Address Undetected.

Bit 2: Data Mark Undetected

If a valid data mark is not detected in the data block of a sector, it is indicated by a Data Mark Undetected error.

Bit 3: Sector Address Undetected

The Sector Address Undetected bit can be set on two conditions; not finding the sector address and a CRC error on an address ID field.

If the disk makes three revolutions during an address search operation and the sector address specified in the sector address register is not found in any of the address ID fields, a Sector Address Undetected condition is indicated.

A CRC error that occurs on an address ID field will set bit 3 also. Table 6 shows how bits 1 and 3 are related.

Table 6 Relationship of CRC Error and Sector Address Undetected

CRC Error (STRB1)	Sector Address Undetected (STRB3)	Condition
0	0	No Error
0	1	Sector Address not Detected
1	0	CRC Error on a Data Field
1	1	CRC Error on Address ID Field

Bit 4: Seek Error

An STZ (Seek Track Zero) command that never receives a track zero indication on the track zero input will result in a Seek Error (see description of STZ command).

Bit 5: File Inoperable

The state of the File Inoperable input appears in bit 5. If the File Inoperable input is a "High" level, a pulse of width equals to Enable pulse width PW_{EL} is issued on the FIR output when STRB is read. FI is not latched but the input is gated to the bus when STRB is read.

Bit 6: Write Error

If the WPT input becomes a "High" level (logic "1") during the execution of a write command the Write Error bit is set.

Bit 7: Hard Error

If the Ready input becomes a "Low" level during the operation of a command (Busy is set), a Hard Error indication will result.

- **General Count Register (GCR); Hex address 5, write only**

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Not Used	7 Bit Count for Track Number on SEK Command and Sector Count for MSR or MSW Command						

The GCR contains the destination track address for the R/W head on an SEK Macro Command. The contents of the GCR are transferred to the CTAR at the end of the SEK Command. For multi-sector read or write operations (MSR, MSW), the GCR contains the number of sectors to be read minus one. During the MSR or MSW execution the GCR is decremented after each sector is read or written.

- **CRC Control Register (CCR) ; Hex address 6, write only**

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Not Used						Shift CRC	CRC Enable

The CCR information is used only in the free format commands; for all other commands this register is masked and has no function.

Bit 0: CRC Enable

During an FFW command, CRC Enable is set by software and CRC generation takes effect on the next transfer of data from DOR to DOSR (see figure 32). The CRC generation continues until Shift CRC (CCR bit 1) is set.

For an FFR command, CRC Enable is set by software and CRC generation takes effect on the next data read from DIR. The calculation continues for all data bytes read from DIR until CRC Enable is reset. The bytes read previous to resetting CRC Enable are considered the CRC information bytes and the CRC check is made against them.

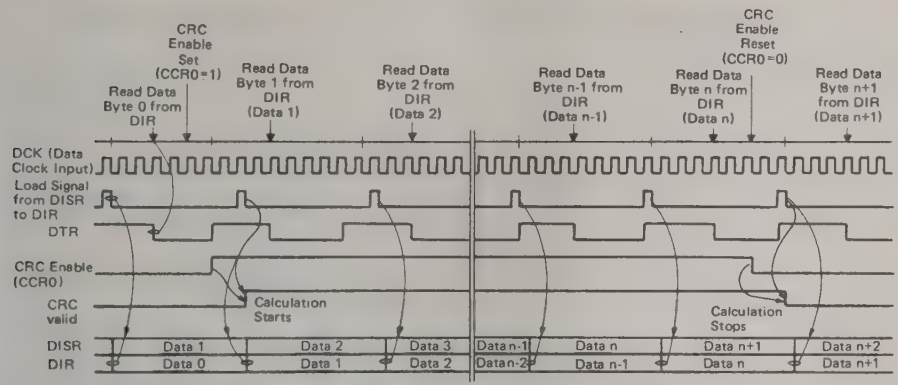
Bit 1: Shift CRC

Bit 1 is valid only for the FFW command. After setting, it takes effect on the next transfer of data from DOR to DOSR (see Figure 33). Setting Shift CRC terminates the CRC calculation and causes the CRC calculated on all the data written into DOR up to the setting of bit 1, to be shifted out the WDT output. The CRC calculation will not include any data written to DOR after Shift CRC is set.

- **LTAR (Logical Track Address); Hex address 7, write only**

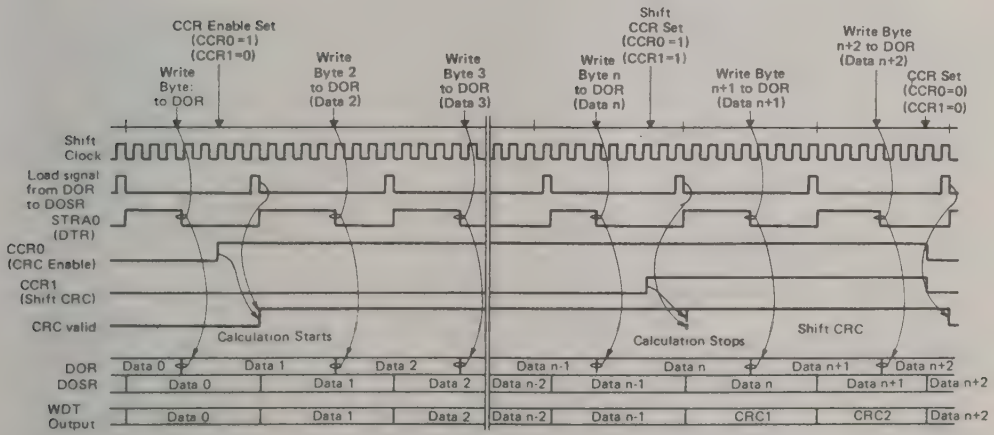
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Not Used	7 Bit Logical Track Address						

When a read or write macro command (SSW, SWD, SSR, RCR, MSW, MSR) is issued, the address of the track on which the operation is to be performed must be written into the LTAR. The address in the track address byte of an Address ID field of the disk is compared with the contents of the LTAR. The contents of LTAR are not affected by the execution of any of the commands.



CRC Calculation includes Data Byte 1 through Data Byte n.

Figure 32 CCR Control Register Timing for an FFR Command (READ)



The CRC Calculation includes Data Byte 1 through Data Byte n-1.

Figure 33 CCR Control Register Timing for an FFW Command (WRITE)

Table 7 Programming Reference Data

Table 7 is a summary of the information in the data sheet and can be used as a reference when programming the HD6843.

Registers	Hex Address	R/W Mode	Data Bits							
			Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
DOR	0	WO	8 Bits of Data Used for a Disk Write Operation							
			Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
DIR	0	RO	8 Bits of Data Used for a Disk Read Operation							
			Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CTAR	1	R/W	Track Address of Current Head Position							
			Bit 7	Bit 6	Bit 5	Bit 4	Bit 3 *	Bit 2 *	Bit 1 *	Bit 0 *
CMR	2	WO	Function Interrupt Mask	ISR3 Interrupt Mask	DMA Flag	FWF	Macro Command			
			Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2 *	Bit 1 *	Bit 0 *
ISR	2	RO	Not Used				STRB -OR	Status Sense Request	Seek Command End	Read Write Command End
			Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SUR	3	WO	Track to Track Seek Time				Head Settling Time			
			Bit 7 *	Bit 6	Bit 5 *	Bit 4	Bit 3	Bit 2	Bit 1 *	Bit 0 *
STRA	3	RO	Busy	Index	Track Not Equal	Write Protect	Track Zero	Drive Ready	Delete Data Mark Detected	Data Transfer Request
			Bit 7	Bit 6	Bit 5	Bit 4 *	Bit 3 *	Bit 2 *	Bit 1 *	Bit 0 *
SAR	4	WO	Not Used			5 Bit Sector Address				
			Bit 7 *	Bit 6 *	Bit 5	Bit 4 *	Bit 3 *	Bit 2 *	Bit 1 *	Bit 0 *
STRB	4	RO	Hard Error	Write Error	File Inoperable	Seek Error	Sector Address Undetected	Data Mark Undetected	CRC Error	Data Transfer Error
			Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
GCR	5	WO	Not Used	7 Bit Count for Track Number on SEK or Sector Count for MSR or MSW.						
			Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CCR	6	WO	Not Used						Shift CRC	CRC Enable
			Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
LTAR	7	WO	Not Used	7 Bit Logical Track Address						

RO — Read Only
WO — Write Only
R/W — Read/Write

* Cleared by $\overline{RES}$

MACRO COMMANDS

Hex Code	Instruction	Hex Code	Instruction
2	STZ	A	FFR
3	SEK	B	FFW
4	SSR	C	MSR
5	SSW	D	MSW
6	RCR		
7	SWD		

Table 8 Error Condition, Command Execution, Interrupt, and Head Control

Error	Flag	Set Condition	Reset Condition	Command	Command Execution	Interrupt	Head Control
Track Not Equal	STRA5	Track information of ID field is not equal to the content of LTAR.	Issuing of SSR, RCR, MSR, SSW, SWD or MSW Command	SSR, RCR, MSR SSW, SWD, MSW	The execution of a command is interrupted and R/W Command End (ISR0) is set.	Request (ISR0)	Unchanged**
Data Transfer Error	STRB0	Overrun or underflow during the data transfer	Reading of STRB	SSR, MSR, SSW, SWD, MSW, FFR FFW	Read/Write command continues to be executed.	No interrupt	Unchanged**
CRC Error	STRB1	CRC Error on ID field or Date field	Reading of STRB	SSR, RCR, MSR, SSW, SWD, MSW (FFR)	The execution of a command is interrupted and R/W Command End (ISR0) is set.	Request (ISR0, ISR3)	Unchanged**
Data Mark Undetected	STRB2	DAM or DDAM is undetected within 32 bytes after ID field has been detected.	Reading of STRB	SSR, RCR, MSR	The execution of a command is interrupted and R/W Command End (ISR0) is set.	Request (ISR0, ISR3)	Unchanged**
Sector Address Undetected	STRB3	(1) Sector Address of ID field is not equal to the content of SAR. (2) CRC Error on ID field	Reading of STRB after Busy (STRA7) is reset.	SSR, RCR, MSR SSW, SWD, MSW	The execution of a command is interrupted and R/W Command End (ISR0) is set.	Request (ISR0, ISR3)	Unchanged (Head remains loaded after settling time has expired.)
Seek Error	STRB4	TRZ signal remains "Low" level though eighty-two STP pulse outputs are provided in STZ command.	Reading of STRB	STZ	The execution of a command is interrupted and Seek Command End (ISR1) is set.	Request (ISR1, ISR3)	Unchanged
File Inoperable	STRB5	A "High" level input of FI terminal is reflected.	FI signal of the FDD is reset when "High" pulse output is provided by reading of STRB at FI="1".	All commands	The execution of a command is interrupted. If it is a Read/Write command, ISR0 is set. If it is a seek command, ISR1 is set.	Request (ISR0 or ISR1, ISR3)	Unload the head imediately (HLD="Low") Set WGT to "Low"
Write Error	STRB6	Write operation (WGT="High") is performed when the input of WPT terminal is "High" level.	Reading of STRB	SSW, SWD, MSW FFW	The execution of a command is interrupted and R/W Command End (ISR0) is set.	Request (ISR0, ISR3)	Unload the head imediately (HLD="Low" Set WGT to "Low"
Hard Error	STRB7	RDY input signal becomes "Low" level during the execution of a command (Busy="1".)	Reading of STRB	All commands	The execution of a command is interrupted. If it is a Read/Write command, ISR0 is set. If it is a seek command, ISR1 is set.	Request (ISR0 or ISR1, ISR3)	Unload the head imediately (HLD="Low") Set WGT to "Low"
Not Ready during the idling	STRA2	—	—	—	—	No interrupt	Unload the head imediately (HLD="Low")

* These errors except STRB5 and STRA2 are reset by $\overline{RES}$ inputs.

** Head is unloaded if the new command is not issued during the settling time after Read/Write command ends.

HD6844, HD68A44, HD68B44

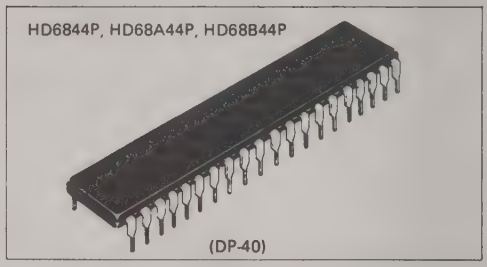
DMAC (Direct Memory Access Controller)

The HD6844 Direct Memory Access Controller (DMAC) performs the function of transferring data directly between memory and peripheral device controllers. It controls the address and data buses in place of the MPU in bus organized systems such as the HMCS6800 Microprocessor System.

The bus interface of the HD6844 includes select, read/write, interrupt, transfer request/grant, and bus interface logic to allow the data transfer over an 8-bit bidirectional data bus. The functional configuration of the DMAC is programmed via the data bus. The internal structure provides for control and handling of four individual channels, each of which is separately configured. Programmable control registers provide control for the transfer location and length, individual channel control and transfer mode configuration, priority of servicing, data chaining, and interrupt control. Status and control lines provide control to the peripheral controllers.

The mode of transfer for each channel can be programmed as cycle-stealing or a burst transfer mode.

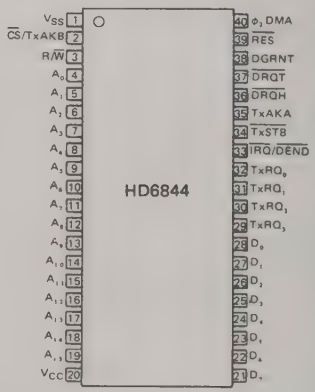
Typical applications would be with the Floppy Disk Controller (FDC), etc..



■ FEATURES

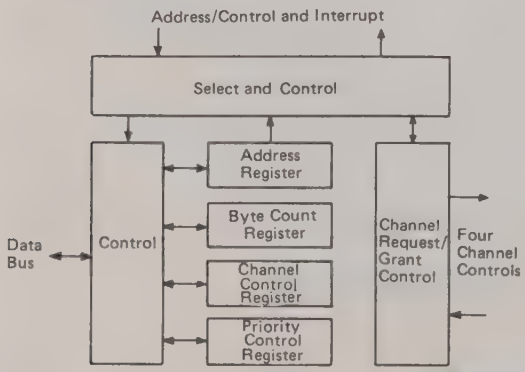
- Four DMA Channels, Each Having a 16-Bit Address Register and a 16-Bit Byte Count Register
- 1 M Byte/Sec (HD6844), 1.5 M Byte/Sec (HD68A44), 2.0 M Byte/Sec (HD68B44) Maximum Data Transfer Rate
- Selection of Fixed or Rotating Priority Service Control
- Separate Control Bits for Each Channel
- Data Chain Function
- Address Increment or Decrement Update
- Programmable Interrupts and DMA End to Peripheral Controllers
- Compatible with MC6844, MC68A44, MC68B44

■ PIN ARRANGEMENT



(Top View)

■ BLOCK DIAGRAM



HITACHI

■ ABSOLUTE MAXIMUM RATINGS

Item	Symbol	Value	Unit
Supply Voltage	V_{CC}*	-0.3 ~ +7.0	V
Input Voltage	V_{in}*	-0.3 ~ +7.0	V
Operating Temperature	T_{opr}	-20 ~ +75	°C
Storage Temperature	T_{stg}	-55 ~ +150	°C

* With respect to V_{SS} (SYSTEM GND)

(NOTE) Permanent LSI damage may occur if maximum ratings are exceeded. Normal operation should be under recommended operating conditions. If these conditions are exceeded, it could affect reliability of LSI.

■ RECOMMENDED OPERATING CONDITIONS

Item	Symbol	min	typ	max	Unit
Power Supply Voltage	V_{CC}*	4.75	5.0	5.25	V
Input Voltage	V_{IL}*	-0.3	–	0.8	V
	V_{IH}*	2.0	–	V_{CC}	V
Operating Temperature	T_{opr}	-20	25	75	°C

* With respect to V_{SS} (SYSTEM GND)

■ ELECTRICAL CHARACTERISTICS (V_{CC}=5V±5%, V_{SS}=0V, Ta=-20~+75°C, unless otherwise noted.)

● DC CHARACTERISTICS

Item		Symbol	Test Condition	min	typ*	max	Unit
Input "High" Voltage		V_{IH}		2.0	–	V_{CC}	V
Input "Low" Voltage		V_{IL}		-0.3	–	0.8	V
Input Leakage Current	TxRQ$_0$~$_3$, ϕ_2 DMA, $\overline{RES}$, DGRNT	I_{in}	V_{in}=0~5.25V	-2.5	–	2.5	μA
Three-State (off state) Leakage Current	A_0~A_{15}, D_0~D_7, R/$\overline{W}$	I_{TSI}	V_{in}=0.4~2.4V	-10	–	10	μA
Output "High" Voltage	D_0~D_7	V_{OH}	I_{OH}=-205μA	2.4	–	–	V
	A_0~A_{15}, R/$\overline{W}$		I_{OH}=-145μA	2.4	–	–	
	All Other Outputs		I_{OH}=-100μA	2.4	–	–	
Output "Low" Voltage		V_{OL}	I_{OL}=1.6mA	–	–	0.4	V
Source Current	$\overline{CS}$/TxAKB	I_{CSS}	V_{in}=0V, Fig. 10	–	10	16	mA
Power Dissipation		P_D		–	500	1000	mW
Input Capacitance	ϕ_2 DMA	C_{in}	V_{in}=0V, T_a=25°C f=1.0MHz	–	–	20	pF
	D_0~D_7, $\overline{CS}$, A_0~A_4, R/$\overline{W}$			–	–	12.5	
	TxRQ$_0$~$_3$, $\overline{RES}$, DGRNT			–	–	10	
Output Capacitance		C_{out}	V_{in}=0V, T_a=25°C, f=1MHz	–	–	12	pF

* V_{CC}=5.0V, T_a=25°C

• AC CHARACTERISTICS (Load Condition Fig. 9)

1. CLOCK TIMING

Item		Symbol	Test Condition	HD6844			HD68A44			HD68B44			Unit
				min	typ	max	min	typ	max	min	typ	max	
ϕ_2 DMA Cycle Time		$t_{cyc\phi}$	Fig. 2	1000	–	–	666	–	–	500	–	–	ns
ϕ_2 DMA Pulse Width	"High" Level	$PW_{\phi H}$	Fig. 2	450	–	–	280	–	–	235	–	–	ns
	"Low" Level	$PW_{\phi L}$	Fig. 2	400	–	–	230	–	–	210	–	–	ns
ϕ_2 DMA Rise and Fall Time		$t_{\phi r}$, $t_{\phi f}$	Fig. 2	–	–	25	–	–	25	–	–	25	ns

2. DMA TIMING (Load Condition Fig. 9)

Item		Symbol	Test Condition	HD6844			HD68A44			HD68B44			Unit
				min	typ	max	min	typ	max	min	typ	max	
TxRQ Setup Time	ϕ_2 DMA Rising Edge	t_{TQS1}	Fig. 3	120	–	–	120	–	–	120	–	–	ns
	ϕ_2 DMA Falling Edge	t_{TQS2}		210	–	–	210	–	–	155	–	–	
TxRQ Hold Time	ϕ_2 DMA Rising Edge	t_{TQH1}		20	–	–	10	–	–	10	–	–	ns
	ϕ_2 DMA Falling Edge	t_{TQH2}		20	–	–	10	–	–	10	–	–	
DGRNT Setup Time	DGRNT	t_{DGS}	Fig. 4	155	–	–	125	–	–	115	–	–	ns
DGRNT Hold Time	DGRNT	t_{DGH}		10	–	–	10	–	–	10	–	–	
Address Output Delay Time	A_0~A_{15}, R/$\overline{W}$, $\overline{TxSTB}$	t_{AD}	Fig. 6	–	–	270	–	–	180	–	–	160	ns
Address Output Hold Time	A_0~A_{15}, R/$\overline{W}$	t_{AHO}	Fig. 6	30	–	–	20	–	–	20	–	–	ns
	$\overline{TxSTB}$		Fig. 7	35	–	–	35	–	–	35	–	–	
Address Three-State Delay Time	A_0~A_{15}, R/$\overline{W}$	t_{ATSD}	Fig. 7	–	–	270	–	–	270	–	–	270	ns
Address Three-State Recovery Time	A_0~A_{15}, R/$\overline{W}$	t_{ATSR}	Fig. 7	–	–	270	–	–	270	–	–	270	ns
Delay Time	$\overline{DRQH}$, $\overline{DRQT}$	t_{DQD}	Fig. 5	–	–	375	–	–	250	–	–	210	ns
TxAK Delay Time	ϕ_2 DMA Rising Edge	t_{TKD1}	Fig. 5	–	–	400	–	–	310	–	–	250	ns
	DGRNT Rising Edge	t_{TKD2}	Fig. 8	–	–	190	–	–	160	–	–	150	
$\overline{IRQ}$/$\overline{DEND}$ Delay Time	ϕ_2 DMA Falling Edge	t_{DED1}	Fig. 6	–	–	300	–	–	250	–	–	210	ns
	DGRNT Rising Edge	t_{DED2}	Fig. 8	–	–	190	–	–	160	–	–	125	

3. BUS TIMING

1) READ TIMING

Item		Symbol	Test Condition	HD6844			HD68A44			HD68B44			Unit
				min	typ	max	min	typ	max	min	typ	max	
Address Setup Time	A_0~A_4, R/$\overline{W}$, $\overline{CS}$	t_{AS}	Fig. 2	140	–	–	140	–	–	70	–	–	ns
Address Input Hold Time	A_0~A_4, R/$\overline{W}$, $\overline{CS}$	t_{AHI}		10	–	–	10	–	–	10	–	–	ns
Data Delay Time	D_7~D_7	t_{DDR}		–	–	320	–	–	220	–	–	180	ns
Data Access Time	D_0~D_7	t_{ACC}		–	–	460	–	–	360	–	–	280	ns
Data Output Hold Time	D_0~D_7	t_{DHR}		10	–	–	10	–	–	10	–	–	ns

2) WRITE TIMING

Item		Symbol	Test Condition	HD6844			HD68A44			HD68B44			Unit
				min	typ	max	min	typ	max	min	typ	max	
Address Setup Time	A_0~A_4, R/$\overline{W}$, $\overline{CS}$	t_{AS}	Fig. 2	140	–	–	140	–	–	70	–	–	ns
Address Input Hold Time	A_0~A_4, R/$\overline{W}$, $\overline{CS}$	t_{AHI}		10	–	–	10	–	–	10	–	–	ns
Data Setup Time	D_0~D_7	t_{DSW}		195	–	–	80	–	–	60	–	–	ns
Data Input Hold Time	D_0~D_7	t_{DHW}		10	–	–	10	–	–	10	–	–	ns

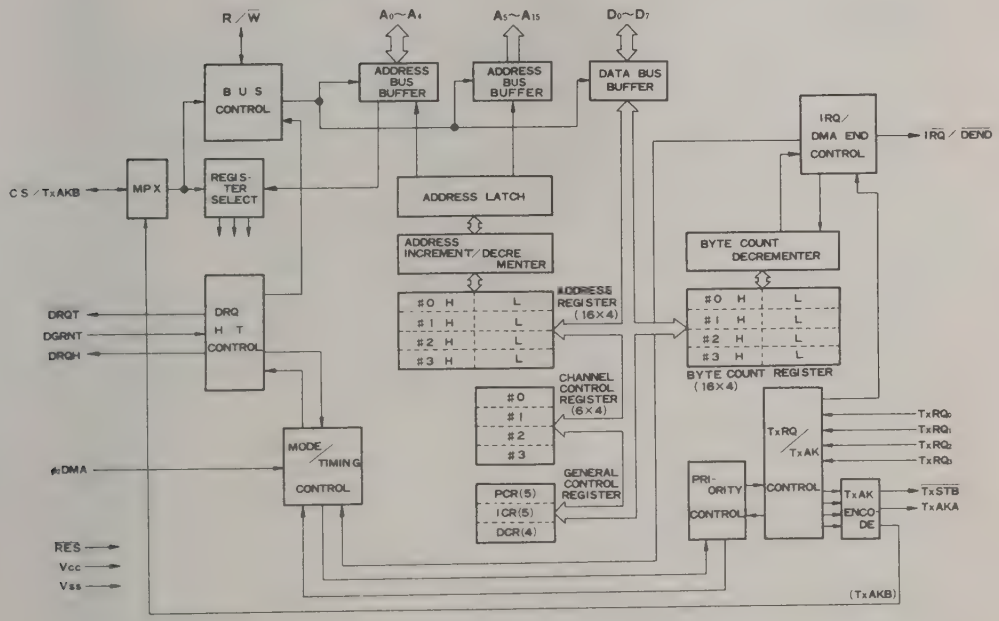


Figure 1 Expanded Block Diagram

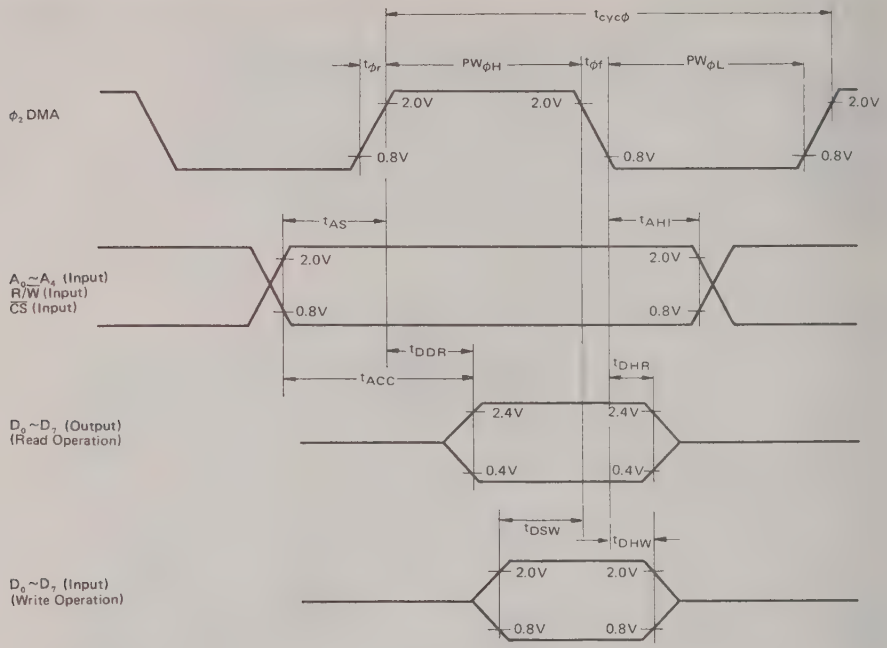


Figure 2 Read/Write Sequence

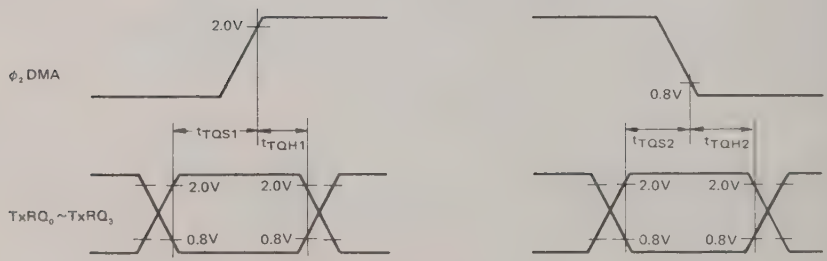
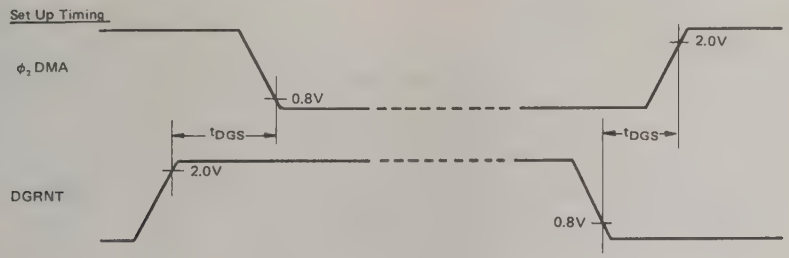


Figure 3 Timing of TxRQ Input



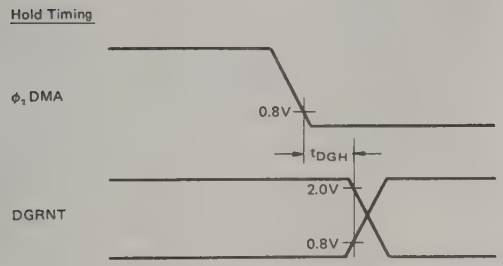


Figure 4 Timing of DGRNT Input

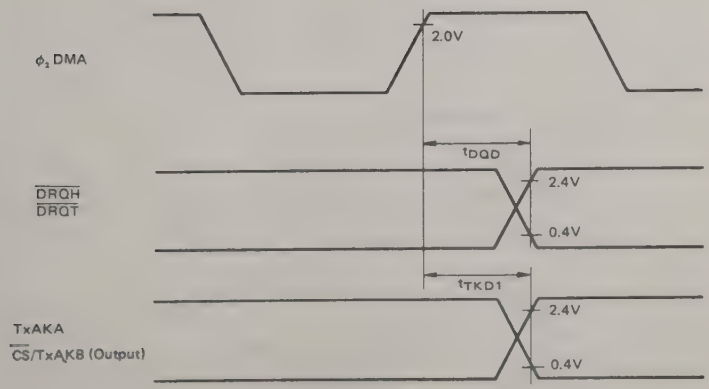


Figure 5 Timing of $\overline{DRQH}$, $\overline{DRQT}$, TxAK Outputs

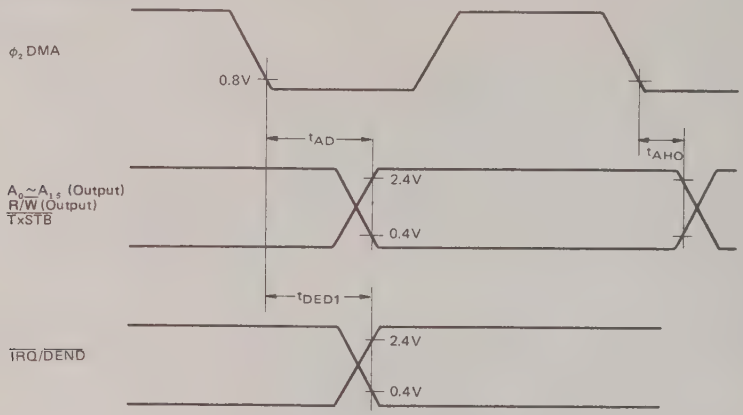
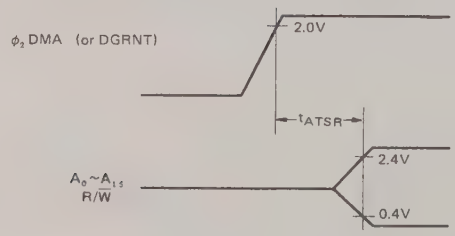


Figure 6 Timing of Address and IRQ/DEND Outputs

Recovery Time of Address Three-state



Delay Time of Address Three-state

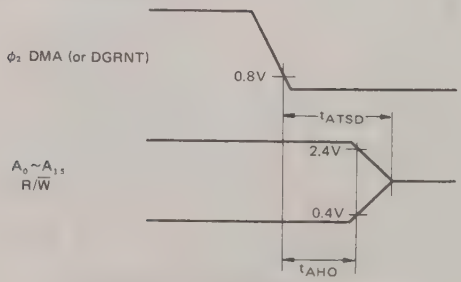


Figure 7 Timing of Address Three-state

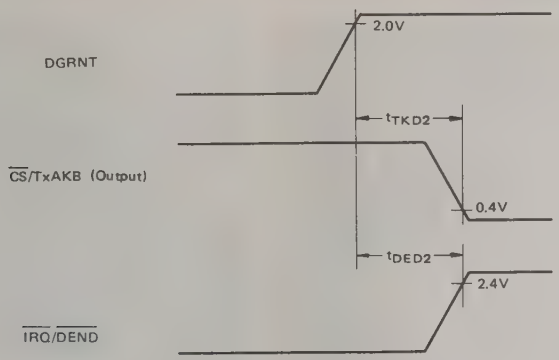
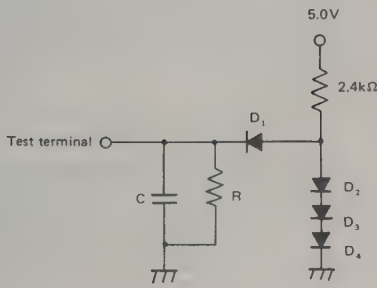


Figure 8 Timing of Synchronous DGRNT Output



Test terminal	C	R
D_0~D_7	130 pF	11 kΩ
A_0~A_{15}, R/$\overline{W}$	90 pF	16 kΩ
$\overline{CS}$/TxAKB	50 pF	24 kΩ
All other outputs	30 pF	24 kΩ

D_1~D_4 : 1S2074 Ⓗ or equivalent.

Figure 9 Load Circuit

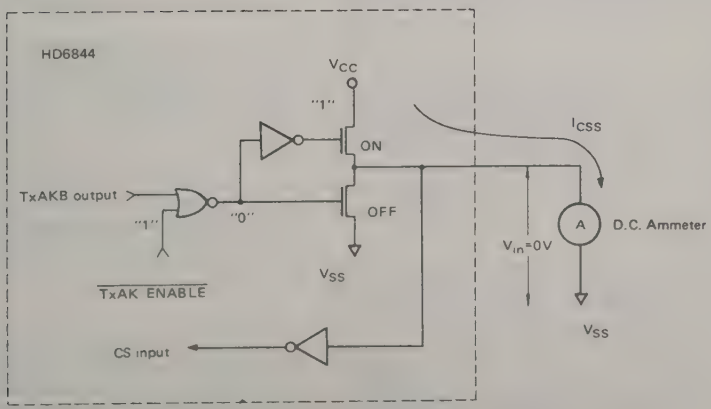


Figure 10 Source Current Measurement Circuit for $\overline{CS}$/TxAKB Terminal

■ DEVICE OPERATION

The DMAC has fifteen addressable registers, eight of them are sixteen bits in length. Each channel has a separate Address Register and a Byte Count Register, each of which is sixteen bits. There are also four Channel Control Registers. The three General Control Registers common to all four channels are the Priority Control Register, the Interrupt Control Register, and the Data Chain Register.

To prepare a channel for DMA, the Address Registers must be loaded with the starting memory address and the Byte Count Register loaded with the number of bytes to be transferred. The bits in the Channel Control Register establish the direction of the transfer, the mode, and the address increment or decrement after each cycle. Each channel can be set for one of three transfer modes: Three-State Control (TSC) Steal, Halt Steal, or Halt Burst. Two read-only status bits in the Channel Control Register indicate when the channel is busy transferring data and when the DMA transfer is completed.

The Priority Control Register enables the transfer requests from the peripheral controllers and establishes either a fixed priority or rotating priority scheme of servicing these requests.

When the DMA transfer for a channel is complete (the Byte Count Register is zero), a $\overline{\text{DMA End}}$ signal is directed to the peripheral controller and an $\overline{\text{IRQ}}$ goes to the MPU. Enabling of these interrupts is done in the interrupt Control Register. The IRQ flag bit is read from this register.

Chaining of data transfers is controlled by the Data Chain Register. When enabled, the contents of the Address and Byte Count Registers for channel #3 are put into the registers of the channel selected for chaining when its Byte Count Register becomes zero. This allows for repetitively reading or writing a block of memory.

During the DMA mode, the DMAC controls the address bus and data bus for the system as well as providing the R/$\overline{\text{W}}$ line and a signal to be used as VMA. When a peripheral device controller desires a DMA transfer, it is requested by a Transfer Request. Assuming this request is enabled and meets the test of highest priority, the DMAC will issue a DMA Request. When the DMAC receives the DMA Grant, it gives a Transfer Acknowledge to the peripheral device controller, at which time the data is transferred. When the channel's Byte Count Register equals zero, the transfer is complete and a $\overline{\text{DMA End}}$ is given to the peripheral device controller, and an $\overline{\text{IRQ}}$ is given to the MPU.

• Initialization

During a power-on sequence, the DMAC is reset via the $\overline{\text{RES}}$ input. All registers, with the exception of the Address and Byte Count Registers, are set to a logic "0" state. This disables all requests and the Data Chain function while masking all interrupts. The Address, Byte Count, and Channel Control Registers must be programmed before the respective transfer request bit is enabled in the Priority Control Register.

• Transfer Modes

There are three ways in which a DMA transfer may be done. The one used is determined by the data transfer rate required, the number of channels attached, and the hardware complexity allowable. Refer to Figures 12, 16 and 17.

Two of the modes, TSC Steal and Halt Steal, are done by cycle-stealing from the MPU. The Three-State Control (TSC) Steal mode is initiated by the DMAC bringing the $\overline{\text{DRQT}}$ line "Low". This line goes to the system clock driver which returns a "High" on DGRNT on the rising edge of the system ϕ_1 clock. The DGRNT signal must cause the address control and data lines to go to the high impedance state. The DMAC now supplies the address from the Address Register of the channel requesting. It also supplies the R/$\overline{\text{W}}$ signal as determined from the Channel Control Register. After one byte is transferred, control is returned to the MPU. This method stretches the ϕ_1 and ϕ_2 clocks while the DMAC uses the memory.

The second method of cycle-stealing is the Halt Steal mode. This method actually halts the MPU instead of stretching the ϕ_1 clock for the transfer period. This mode is initiated by the DMAC bringing the $\overline{\text{DRQH}}$ line "Low". This line connects to the MPU $\overline{\text{HALT}}$ input. The MPU Bus Available (BA) line is the DGRNT input to the DMAC. While the MPU is halted, its Address Bus, Data Bus, and R/$\overline{\text{W}}$ are in the high impedance state. The DMAC now supplies the address and R/$\overline{\text{W}}$ line. After one byte is transferred, the $\overline{\text{HALT}}$ line is returned "High" and the MPU regains control. In this mode, the MPU stops internal activity and is removed from the system while the DMAC uses the memory.

The third mode of transfer is the Halt Burst mode. This mode is similar to the Halt Steal mode, except that the transfer does not stop with one byte. The MPU is halted while an entire block of data is transferred. When the channel's Byte Count Register equals zero, the transfer is complete and control is returned to the MPU. This mode gives the highest data transfer rate, at the expense of the MPU being inactive during the transfer period.

■ INPUT/OUTPUT FUNCTIONS

• DMAC Interface Signals for the MPU

The DMAC interfaces with the HMCS6800 MPU through the eight-bit bidirectional data bus, the $\overline{\text{CS}}$ line, five address lines, an $\overline{\text{IRQ}}$ line, the Read/Write line, and the $\overline{\text{RES}}$ line. These signals, in conjunction with the HMCS6800 VMA output, permit the MPU to have access to the DMAC. Four other lines associated with the MPU and the clock driver are the $\overline{\text{DRQT}}$, $\overline{\text{DRQH}}$, DGRNT, and the ϕ_2 DMA.

Bidirectional Data (D_0~D_7)

The Bidirectional Data lines (D_0~D_7) allow for data transfer between the DMAC and the MPU. The data bus output drivers are three-state devices that remain in the high impedance state except when the MPU performs DMAC read operations.

Chip Select/Transfer Acknowledge B ($\overline{\text{CS}}$/T x AKB)

This line is multiplexed, serving both as an input and an output. $\overline{\text{CS}}$/TxAKB is an output in the four-channel mode during the DMA transfer. At all other times, it is a high impedance TTL compatible input used to address the DMAC. The DMAC is selected when $\overline{\text{CS}}$/TxAKB is "Low". VMA must be used in generating this input to insure that false selects will not occur. Transfers of data to and from the DMAC are then performed under the control of the ϕ2 DMA, Read/Write, and A_0~A_4 address lines. In the four-channel mode when TxAKB is needed, the $\overline{\text{CS}}$ gate must have an open-collector output (a pull-up resistor should not be used). In the two-channel mode, $\overline{\text{CS}}$/TxAKB is always an input.

Address Lines (A_0~A_4)

Address lines A_0~A_4 are both input and output lines. In the MPU mode, these are high impedance inputs used to address the DMAC registers. In the DMA mode, these lines are outputs which are set to the contents of the Address Register of the channel being processed.

Interrupt Request/DMA End ($\overline{IRQ}/\overline{DEND}$)

$\overline{IRQ}/\overline{DEND}$ is a TTL compatible, active "Low" output that is used to interrupt the MPU and to signal the peripheral controller that the data block transfer has ended. If the Interrupt has been enabled, the $\overline{IRQ}/\overline{DEND}$ line will go "Low" after the last DMA cycle of a transfer. An open collector gate must be connected to DGRNT and $\overline{IRQ}/\overline{DEND}$ to prevent false interrupts from the DEND signal when interrupts are not enabled. Refer to the section of "DMA End Control".

Read/Write (R/$\overline{W}$)

Read/Write is a TTL compatible line that is a high impedance input in the MPU mode and an output in the DMA mode. In the MPU mode, it is used to control the direction of data flow through the DMAC's input/output data bus interface. When Read/Write is "High" (MPU read cycle) and the chip is selected, DMAC data output buffers are turned on and a selected register is read. When it is "Low", the DMAC output drivers are turned off and the MPU writes into a selected register.

In the DMA mode, Read/Write is an output to drive the memory and peripheral controllers. Its state is determined by bit 0 of the Channel Control Register for the channel being serviced. When Read/Write is "High", the memory is read and the data from the memory is written into the peripheral controller. When it is "Low", the peripheral controller is read and its data stored in the memory. In the DMA mode, the DMAC data buffers are off.

Reset ($\overline{RES}$)

The $\overline{RES}$ input provides a means of resetting the DMAC from an external source. In the "Low" state, the $\overline{RES}$ input causes all registers, with the exception of the Address and Byte Count Registers, to be reset to the logic "0" state. This disables all transfer requests, masks all interrupts, disables the data chain function, and puts each Channel Control Register into the condition of memory write, Halt Steal transfer mode, and address increment.

- **Transfer Signals to the MPU**

Two DMA request output lines and a DMA Grant input line, together with the system clock, synchronize the DMAC with the MPU system.

DMA Request Three-State Control Steal ($\overline{DRQT}$)

This active "Low" output requests a DMA transfer for a channel configured for the TSC Steal transfer mode. This line is connected to the system clock driver, requesting a ϕ_1 clock stretch. It will remain in the "Low" state until the transfer has begun.

DMA Request Halt ($\overline{DRQH}$)

This active "Low" output requests a DMA transfer for a channel programmed for the Halt Steal or Halt Burst mode transfer. This line is connected directly to the MPU $\overline{HALT}$ input and remains "Low" until the last byte has begun to be transferred.

DMA Grant (DGRNT)

This is a high impedance input to the DMAC, giving it control of the system busses. For the TSC Steal mode, the signal comes from the system clock drive circuit (DMA Grant), indicating that the clock is being stretched. For either of the Halt modes, this signal is the Bus Available from the MPU, indicating that the MPU has halted and turned control of its busses over to the DMAC. For a design involving TSC Steal and Halt mode transfers, this input must be the OR of the clock driven DMA Grant and the MPU BA.

ϕ_2 DMA

Transferring in and out of the DMAC registers, sampling of channel request lines and gating of other control signals to the system is done internally in conjunction with the ϕ_2 DMA high impedance input. This input must be the system memory clock (non-stretched ϕ_2 clock).

- **Transfer Signals From the Peripheral Controller**

Transfer Request ($TxRQ_0 \sim TxRQ_3$)

Each of the four channels has its own high impedance input request for transfer line. The peripheral controller requests a transfer by setting its TxRQ line "High" (a logic "1"). The lines are sampled according to the priority and enabling established in the Priority Control Register. In the Steal mode and the first byte of the Halt Burst mode, the TxRQ signals are tested on the positive edge of ϕ_2 DMA and the highest priority channel is strobed. Once strobed, the TxRQs are not tested until that channel's data transfer is finished. In the succeeding bytes of the Halt Burst mode transfer, the TxRQ is tested on the negative edge of ϕ_2 DMA, and data is transferred on the next ϕ_2 DMA cycle if TxRQ is "High".

- **Transfer Signals to the Peripheral Controller**

Two encoded lines select the channel to be serviced. A strobe line acknowledges the request and performs the transfer. The $\overline{DEND}$ line signals to the peripheral controller that the DMA transfer is completed.

Transfer Acknowledge A (T x AKA)

The Transfer Acknowledge A (TxAKA) is a TTL compatible output used in conjunction with the $\overline{CS}$/TxAKB line to select the channel to be strobed for transfer and to give the DMA End Signal. In the two-channel mode, only TxAKA is used to select channel 0 or channel 1, and $\overline{CS}$/TxAKB is always an input.

Chip Select/Transfer Acknowledge B ($\overline{CS}$/TxAKB)

In the DMA mode, this dual purpose line is encoded together with TxAKA to select the channel being serviced. Table 1 shows the encoding order.

Table 1 Encoding Order

$\overline{CS}$/TxAKB	TxAKA	Channel #
0	0	0
0	1	1
1	0	2
1	1	3

Transfer Strobe ($\overline{TxSTB}$)

The $\overline{TxSTB}$ causes acknowledgement to be given to the peripheral controller and transfers the data to or from the memory. This line is also intended to be the VMA signal for the system in the DMA mode. In a one-channel system, $\overline{TxSTB}$ may be inverted and run to the peripheral controller's Acknowledge input. In a two or four-channel system, $\overline{TxSTB}$ enables the decode of TxAKA and $\overline{CS}$/TxAKB to select the device controller to be acknowledged.

Interrupt Request/DMA End ($\overline{IRQ}/\overline{DEND}$)

In the DMA mode, this dual purpose line is "Low" for the last byte of transfer, indicating a DMA End. This occurs when the Byte Count register decrements to zero.

This line, through the decode of TxAKA and $\overline{CS}$/TxAKB, can be used to strobe a DMA End to each device controller.

- **Address Lines to the Memory**

Address Lines ($A_0 \sim A_{15}$)

These output lines are in the high impedance state during the MPU mode. In the DMA mode, these lines are outputs which are set to the contents of the Address Register of the channel being processed.

■ THE DMAC REGISTERS

The HD6844 (DMAC) has Address Register (ADR), Byte Count Register (BCR), Channel Control Register (CHCR), and General Control Register (GCR).

General Control Register (GCR) is composed of Priority Control Register (PCR) that controls priority among the channels, Interrupt Control Register (ICR) that controls interrupt and Data Chain Control Register (DCR) that controls data chain function. Refer to Table 2 and Figure 1.

These are Read/Write registers and MPU can exchange the data with DMAC when $\overline{CS}$ is at "Low" level. $A_0 \sim A_4$ specifies the address of the registers. How to specify the registers is shown in Table 2.

2-byte ADR and BCR can be read or written by one instruction, using 2-byte instruction of the MPU.

- **Function of Internal Registers**

ADR (Address Register)

Each channel has 16-bit Address Register. Initial address of memory used for DMA transfer is programmed to this register. The contents of ADR are output to address bus ($A_0 \sim A_{15}$) during DMA transfer operation. When 1-byte transfer has completed, the 16-bit address is incremented or decremented by one.

The address which the MPU reads out is the renewed one, that is, the memory address for the next transfer. When 1-block transfer has completed, final memory address +1 or -1 is read out.

Table 2 Internal Registers of the DMAC

Register Name	Symbol	Channel	Address Bus Signal					Address (Hexadecimal)
			A_4	A_3	A_2	A_1	A_0	
Address Register	ADRH	0	0	0	0	0	0	00
	ADRL	0	0	0	0	0	1	01
Byte Count Register	BCRH	0	0	0	0	1	0	02
	BCRL	0	0	0	0	1	1	03
Address Register	ADRH	1	0	0	1	0	0	04
	ADRL	1	0	0	1	0	1	05
Byte Count Register	BCRH	1	0	0	1	1	0	06
	BCRL	1	0	0	1	1	1	07
Address Register	ADRH	2	0	1	0	0	0	08
	ADRL	2	0	1	0	0	1	09
Byte Count Register	BCRH	2	0	1	0	1	0	0A
	BCRL	2	0	1	0	1	1	0B
Address Register	ADRH	3	0	1	1	0	0	0C
	ADRL	3	0	1	1	0	1	0D
Byte Count Register	BCRH	3	0	1	1	1	0	0E
	BCRL	3	0	1	1	1	1	0F
Channel Control Register	CHCR	0	1	0	0	0	0	10
	CHCR	1	1	0	0	0	1	11
	CHCR	2	1	0	0	1	0	12
	CHCR	3	1	0	0	1	1	13
Priority Control Register	PCR	–	1	0	1	0	0	14
Interrupt Control Register	ICR	–	1	0	1	0	1	15
Data Chain Control Register	DCR	–	1	0	1	1	0	16

(NOTE) 1) All the registers can be accessed by Read/Write operation. Unused bit of the register is read out "0".
2) H/L of ADR and BCR means the higher (H) 8 bits/the lower (L) 8 bits of a 16-bit register.
3) Being allocated to continuous address, 16-bit ADR and BCR can be read or written by one instruction, using MPU's 2-byte LOAD/STORE instruction.

e.g. LDX $ Register Address of DMAC

(ADRH 3) → (Index Register H)
(ADRL 3) → (Index Register L)

BCR (Byte Count Register)

Each channel has a 16-bit Byte Count Register. Number of DMA transfer words is programmed into this register. The content of the Byte Count Register is decremented by one everytime one-byte transfer has completed. When it becomes "0", $\overline{DEND}$ output goes "Low" level and informs I/O controller of the end of one-block DMA transfer. When $\overline{IRQ}$ is not masked, $\overline{IRQ}$ output goes "Low" level and MPU is interrupted to be informed of the end of DMA transfer. Moreover, $\overline{IRQ}$ and $\overline{DEND}$ signals are output, multiplexed with $\overline{IRQ}/\overline{DEND}$ pin.

CHCR (Channel Control Register)

Each channel has Channel Control Register. This register is used to program the control information of its corresponding channel. Structure of CHCR is shown in Table 3.

(1) R/W Control (specifies the direction of transfer)
Bit – CHCR Bit 0

This bit controls the direction of DMA transfer. When it is at "1", R/$\overline{W}$ signal of DMAC goes "High" level during DMA transfer operation. This means to read out memory and write into I/O controller, that is, data is transferred from memory to I/O controller.

When it is at "0", R/$\overline{W}$ output goes "Low" level and data is transferred from I/O controller to memory.

Table 3 Bit Structure of CHCR (Channel Control Register)

Bit No.	Name	Read/Write	Function	
			"1"	"0"
0	R/W	R/W	Transfer from memory to I/O controller (R/$\overline{W}$ output = "High")	Transfer from I/O controller to memory (R/$\overline{W}$ output = "Low")
1	Burst/Cycle Steal	R/W	Burst Mode	Cycle Steal Mode*
2	TSC/HALT	R/W	TSC Mode	HALT Mode*
3	Address down/up	R/W	Address: -1	Address: +1
4	Not used	–	–	–
5	Not used	–	–	–
6	Busy/Ready Flag	R	Busy (DMA Transfer Operation)	Ready (No DMA Transfer Operation)
7	DEND Flag	R	DMA End & Interrupt	No Interrupt

* Burst transfer in TSC mode is prohibited. R: Read, W: Write

Note that during DMA transfer operation, the function of R/$\overline{W}$ signal is accommodated to the memory Read/Write operation. Therefore, on the side of I/O device during DMA transfer operation, R/$\overline{W}$ input should be interpreted in inverse of the MPU Read/Write. That is, data should be output when R/$\overline{W}$ input is at "Low" level (In the case of MPU's read operation, I/O device outputs the data when it is at "High" level).

This arises from that during DMA transfer operation, I/O side performs data transfer independently instead of MPU. Moreover, such family LSI as HD6843 (FDC), etc. has this function and R/$\overline{W}$ signal is automatically interpreted inversely.

(2) Burst/Cycle Steal Bit – CHCR Bit 1

This bit is used to decide that DMA transfer should be performed in burst mode or cycle steal mode. When it is at "1", it specifies burst mode. That is, once DMA transfer is performed, MPU remains stopped until one-block data transfer is completed.

When this bit is "0", it specifies cycle steal mode. That is, everytime one-byte transfer has completed, MPU takes back the bus control, and DMA transfer and MPU operation are performed in time sharing.

(NOTE) Only in the case of HALT mode, burst mode can be specified. In TSC mode, burst mode cannot be specified.

(3) TSC/HALT Mode Bit – CHCR Bit 2

This bit is used to decide that DMA transfer should be performed by using MPU's TSC function or HALT function. When it is at "0", DMA transfer request signal is output from $\overline{DRQH}$ of DMAC.

When it is at "1", DMA transfer request signal is output from $\overline{DRQT}$ of DMAC.

(4) Address down/up Bit – CHCR Bit 3

This bit is used to decide that the address of memory region used for DMA transfer should be renewed up (increment of address) or down (decrement of address). When it is at "1", the address is decremented by one after one-byte transfer. When it is at "0", the address is incremented by one.

(5) Busy/Ready Flag Bit – CHCR Bit 6

This bit is a status flag to indicate whether its corresponding channel is performing DMA transfer or not. (READ only)

When it receives the first TxRQ of its corresponding channel, it goes to "1". When one-block transfer is completed and BCR becomes "0", it is reset to "0".

Also this flag is cleared when corresponding TxRQ Enable Bit in the PCR becomes "0".

(6) DEND Flag Bit – CHCR Bit 7

This bit is an interrupt flag to indicate that one-block DMA transfer of its corresponding channel has completed. (READ only).

When one-block transfer of its corresponding channel is completed and BCR becomes "0", it goes to "1". As soon as this flag is read out, i.e. CHCR of this channel is read

out, it is reset to "0".

Moreover, this bit is connected to $\overline{IRQ}$ output. When it is at "1" and IRQ enable bit (within ICR register described later) is at "1", $\overline{IRQ}$ output goes "Low" level.

PCR (Priority Control Register)

Priority Control Register is a 5-bit register to decide the operation mode of priority control circuit. Structure of PCR is shown in Table 4.

Table 4 Bit Structure of PCR (Priority Control Register)

Bit No.	Name	Read /Write	Function	
			"1"	"0"
0	TxRQ Enable #0 ($TxEN_0$)	R/W	TxRQ of Channel 0 is accepted.	TxRQ of Channel 0 is not accepted.
1	TxRQ Enable #1 ($TxEN_1$)	R/W	TxRQ of Channel 1 is accepted.	TxRQ of Channel 1 is not accepted.
2	TxRQ Enable #2 ($TxEN_2$)	R/W	TxRQ of Channel 2 is accepted.	TxRQ of Channel 2 is not accepted.
3	TxRQ Enable #3 ($TxEN_3$)	R/W	TxRQ of Channel 3 is accepted.	TxRQ of Channel 3 is not accepted.
4	Not used	–	–	–
5		–	–	–
6		–	–	–
7	Rotate Control	R/W	Rotate Mode	The order of priority is fixed at numerical order.

R: Read, W: Write

(1) TxRQ Enable Bit ($TxEN_0$~$TxEN_3$) – PCR Bit 0~3

Each channel has this TxRQ Enable bit. When it is at "1", TxRQ input of its corresponding channel is accepted to perform DMA transfer. When it goes to "0", TxRQ of its corresponding channel is masked not to be received and TxAK is not output. During DMA transfer operation, when this bit goes to "0" before BCR becomes "0", following TxRQ input is not accepted and DMA transfer is interrupted. Then contents of ADR and BCR remain unchanged. When it rises to "1" again, DMA transfer is reopened. Therefore, in the case of cycle steal DMA, it is possible for the program to change the priority of the specific channel temporarily by manipulating this bit.

(2) Rotate Control Bit – PCR Bit 7

When this bit is at "0", the order of priority among DMA channels is fixed at numerical order. That is, Channel 0 is given a first priority and then is followed by Channel $1 \rightarrow 2 \rightarrow 3$.

When this bit is at "1", priority control is due to rotate mode. That is, the channel that ended in the first time is given a first priority and the channel ended in the last time is controlled to be given a last priority.

ICR (Interrupt Control Register)

Interrupt Control Register is a 5-bit register to control $\overline{IRQ}$ output. Its structure is shown in Table 5.

(1) IRQ Enable Bit – ICR Bit 0~3

Each channel has IRQ Enable Bit. When this bit is at "1" and DEND Flag of its corresponding channel is set to "1", $\overline{IRQ}$ output goes "Low" level. But when it is at "0", $\overline{IRQ}$ output is masked not to be output even if DEND Flag is set to "1".

These bits enable to control to output only a necessary channel to $\overline{IRQ}$.

(2) IRQ Flag – ICR Bit 7

This is a read-only bit and the status of $\overline{IRQ}$ output is directly reflected on it. That is, when $\overline{IRQ}$ output goes to "Low" level, it becomes "1".

$\overline{IRQ}$ output of DMAC is output as logical OR of 4-channel DEND Flag according to the following equation.

$$\overline{IRQ} = (DEND_0 \cdot IRQ\ Enable_0) + (DEND_1 \cdot IRQ\ Enable_1) + (DEND_2 \cdot IRQ\ Enable_2) + (DEND_3 \cdot IRQ\ Enable_3)$$

DCR (Data Chain Control Register)

Data Chain Control Register is a 4-bit register and three of those bits are used to control data chain function. Remaining one bit is used to specify 2-channel/4-channel mode.

Structure of DCR is shown in Table 6.

(1) Data Chain Enable Bit – DCR Bit 0

When this bit is at "1", data chain function of DMAC is enabled. That is, when DMA transfer of a specified channel has completed and BCR goes to "0", the contents of ADR and BCR of Channel #3 are automatically transferred to ADR and BCR of the specified channel.

(2) Data Chain Channel Bit – DCR Bit 1~2

These bits are used to specify which channel should be used for the data chain. How to specify the channel is shown in Table 7. Data Chain Channel bit specifies the channel to which data should be transfered from Channel #3. Channel #3 contains the data for replacement. Channel #3 is fixed and cannot be changed.

(3) 2/4-channel Mode Bit – DCR Bit 3

This bit has no relation to the data chain function.

It is used to specify whether $\overline{CS}$/TxAKB is used for only input pin or I/O pin. When this bit is "0", $\overline{CS}$/TxAKB becomes $\overline{CS}$ input pin in 2-channel mode since TxAKB output is not necessary for application up to 2-channel.

When this bit is "1", $\overline{CS}$/TxAKB becomes I/O pin in 4-channel mode (See Fig. 11).

Table 5 ICR (Interrupt Control Register)

Bit No.	Name	Read /Write	Function	
			"1"	"0"
0	IRQ Enable #0	R/W	$\overline{IRQ}$ of Channel 0 is able to be output.	$\overline{IRQ}$ output of Channel 0 is masked.
1	IRQ Enable #1	R/W	$\overline{IRQ}$ of Channel 1 is able to be output.	$\overline{IRQ}$ output of Channel 1 is masked.
2	IRQ Enable #2	R/W	$\overline{IRQ}$ of Channel 2 is able to be output.	$\overline{IRQ}$ output of Channel 2 is masked.
3	IRQ Enable #3	R/W	$\overline{IRQ}$ of Channel 3 is able to be output.	$\overline{IRQ}$ output of Channel 3 is masked.
4	Not used	–	–	–
5		–	–	–
6		–	–	–
7	IRQ Flag	R	$\overline{IRQ}$ output "Low"	$\overline{IRQ}$ output "High" (off state)

R: Read, W: Write

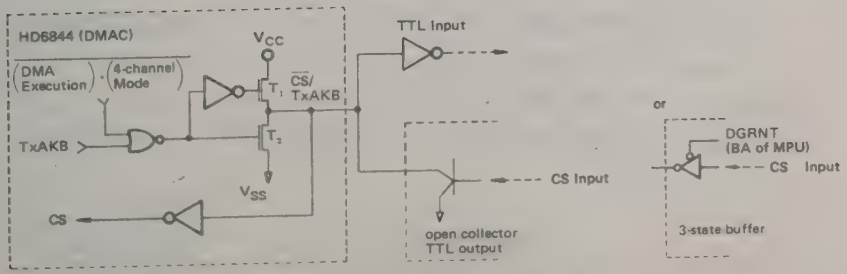
Table 6 Bit Structure of DCR (Data Chain Control Register)

Bit No.	Name	Read /Write	Function	
			"1"	"0"
0	Data Chain Enable	R/W	Data Chain is performed.	Data Chain is not performed.
1	Data Chain Channel	R/W	The channel which performs Data Chain is specified. (The channel where contents of ADR and BCR of Channel #3 are loaded.)	
2		R/W		
3	2/4-Channel Mode	R/W	4-Channel Mode ($\overline{CS}$/TxAKB is I/O pin.)	2-Channel Mode ($\overline{CS}$/TxAKB is designated to only input pin.)
4	Not used	–	–	–
5		–	–	–
6		–	–	–
7		–	–	–

R: Read, W: Write

Table 7 How to specify Data Chain Channel

DCR Bit 1	DCR Bit 2	Specified Channel
0	0	Channel #0
1	0	Channel #1
0	1	Channel #2
1	1	–



In $\overline{CS}$ input mode T1 turns ON and T2 turns OFF. T1 functions as pull-up resistance.

Figure 11 How to Use CS/TxAKB Pin

■ OPERATION OF THE DMAC

● Transfer Mode of the DMAC

There are three DMA transfer modes such as HALT Cycle Steal, HALT Burst and TSC Cycle Steal. Operation in each mode is explained in the following.

HALT Cycle Steal Mode

This is a basic DMA transfer mode utilizing HALT state of MPU. In this mode, everytime 1-byte transfer has completed, MPU takes back the bus control and executes instruction cycle. That is, DMA transfer and MPU operation are performed in time sharing.

Timing chart is shown in Fig. 12 and flow chart is shown in Fig. 13. Procedure of transfer operation is the following. (No. ① ~ ⑪ in Fig. 12 correspond to the following items.)

① $TxRQ_0$~$TxRQ_3$ input is checked at the rising edge of ϕ2DMA. When it is at "High" level, it gets into the following operation.
② $\overline{DRQH}$="Low" is output and MPU is requested to stop its operation.
③ TxAKA is driven (Level output).
④ MPU stops its operation and DMAC waits until DGRNT goes to "High" level.
⑤ When DGRNT goes to "High" level, DMAC drives TxAKB, A_0~A_{15} and R/$\overline{W}$ lines.
⑥ $\overline{TxSTB}$ is given to perform DMA transfer.
⑦ Address is incremented or decremented by one and number of transfer words is decremented by one.
⑧ When $\overline{DRQH}$ rises to "High" level, MPU gets into Instruction Cycle again.
⑨ TxRQ falls to "Low" level.
⑩ A_0~A_{15} and R/$\overline{W}$ get into high impedance state again.
⑪ DGRNT falls to "Low" level.

[Note] $TxRQ_0$~$TxRQ_3$ input in HALT cycle steal is, in principle as shown in Fig. 12, set to "High" every 1-byte transfer on account of I/O request. When $\overline{TxSTB}$ of the DMAC is driven, it is reset to "Low". Take care not to be against this principle, or the following states may happen.

(1) In the case where TxRQ becomes "High", but it is reset to "Low" before DGRNT becomes "High". In this case, the DMAC is in the wait state without sending out $\overline{TxSTB}$ until TxRQ rises to "High" again. As $\overline{DRQH}$ remains "Low" the MPU is forced to be stopped, and the system is in dead lock state until TxRQ rises to "High" again (Fig. 14).

(2) In the case where TxRQ is not reset to "Low" though $\overline{TxSTB}$ has been driven.
In this case, unless TxRQ returns to "Low" by the time ϕ_2DMA rises after $\overline{TxSTB}$ has risen to "High", it is considered as a new I/O request, which leads the above-mentioned operation ①,② ⟶ .
If TxRQ falls to "Low" immediately after that, the same state as (1) happens (Fig. 15).

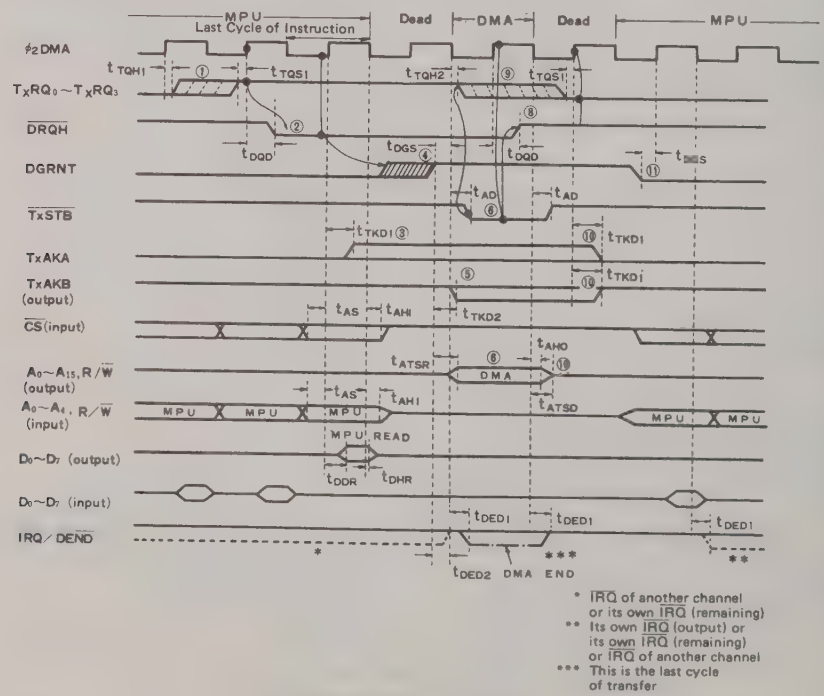


Figure 12 HALT Cycle Steal Mode

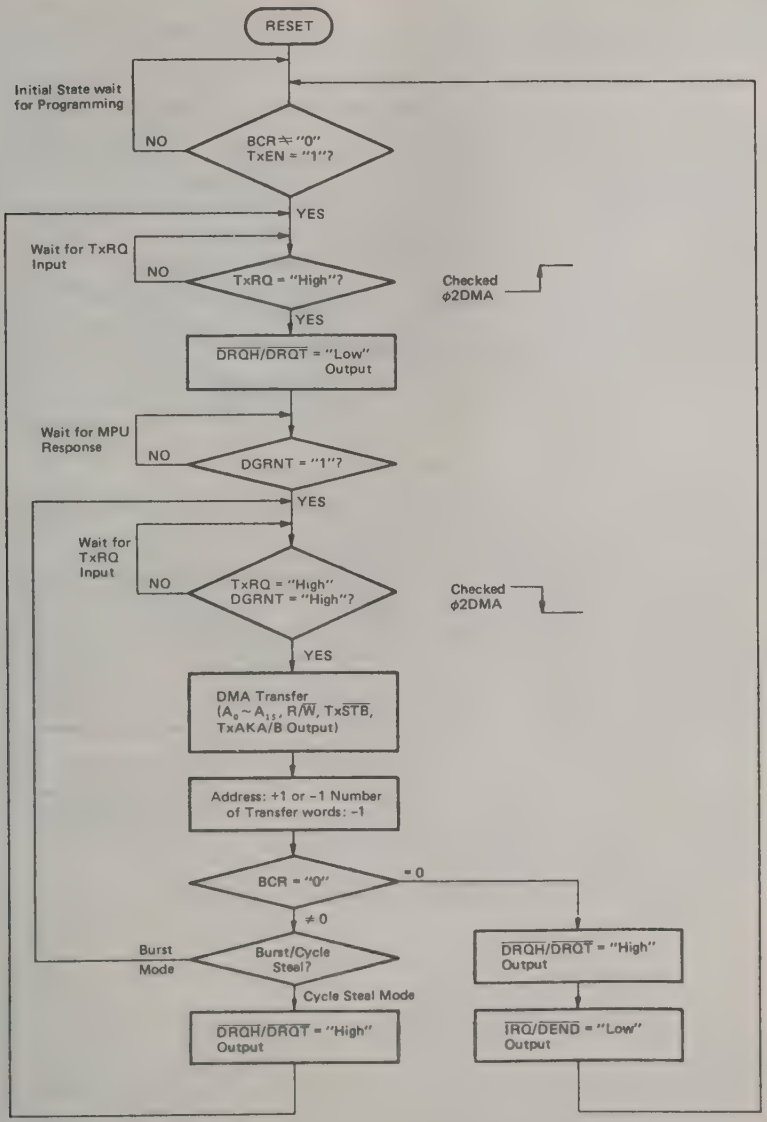


Figure 13 Flow Chart of DMAC Operation

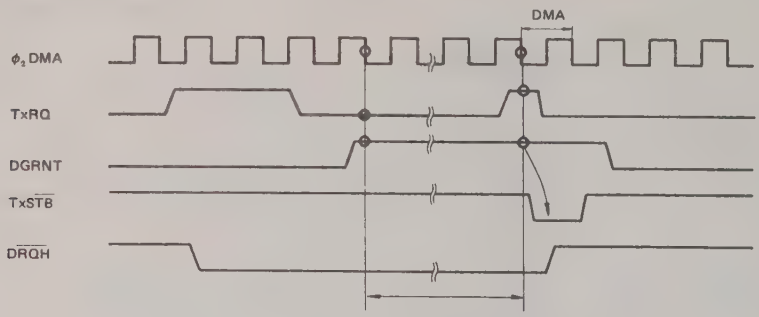


Figure 14 Extraordinary TxRQ Input (1)

(In the case where TxRQ is reset to "Low" before the transfer)

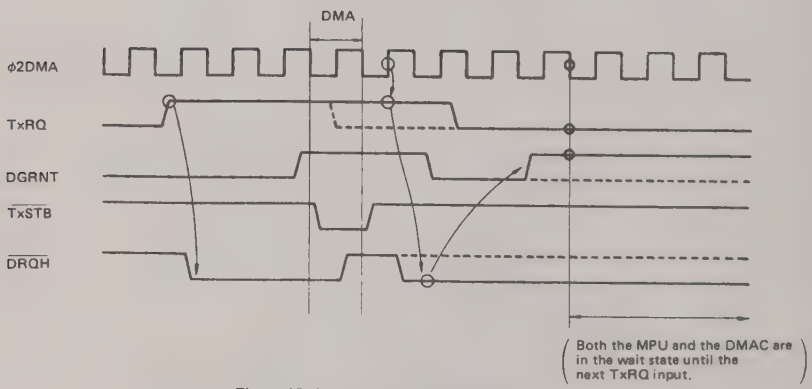


Figure 15 Extraordinary TxRQ Input (2)

(In the case where TxRQ doesn't fall to "Low" after the transfer has been completed.)

HALT Burst Mode

In the case of cycle steal mode, MPU gets into Instruction Cycle everytime 1-byte transfer has completed. But in the case of burst mode, MPU remains stopped until 1-block transfer is finished. That is, $\overline{DRQH}$ continues to be output "Low" level until BCR becomes "0".

Its timing chart and flow chart are shown in Fig. 16 and Fig. 13 respectively. Procedure of transfer is the following (No. ① ~ ⑭ in Fig. 16 correspond to the following items).

① TxRQ input is checked at the rising edge of ϕ_2 DMA. When it is at "High" level, it gest into the following operation.
② $\overline{DRQH}$="Low" level is given and MPU is requested to stop its operation.
③ TxAKA is driven.
④ MPU stops and DMAC waits for DGRNT rising "High" level.
⑤ When DGRNT rises "High" level, DMAC drives TxAKB, A_0 ~A_{15}, and R/$\overline{W}$ lines.
⑥ $\overline{TxSTB}$ is sent out to perform DMA transfer.
⑦ Address is incremented or decremented by one and number of transfer words is decremented by one.
⑧ TxRQ falls to "Low" level.
⑨ When number of transfer words is 0, from ⑪ to ⑭ operations are performed.
⑩ When BCR is not "0", TxRQ is checked at the falling edge of ϕ_2 DMA. When TxRQ is at "High" level, DMA transfer is performed through ⑥ ~ ⑧ again. When TxRQ is not at "High" level, DMAC waits for becoming "High" level.
⑪ $\overline{IRQ}$/$\overline{DEND}$ output goes to "Low" level.
⑫ $\overline{DRQH}$ output rises to "High" level and MPU gets into Instruction Cycle again.
⑬ A_0~A_{15} and R/$\overline{W}$ get into high impedance state.
⑭ DGRNT falls to "Low" level.

The transfer of the first byte (①~⑥) is performed in the same way as that in HALT cycle steal mode. But in the second-byte and subsequent transfer, TxRQ is checked at the falling edge of ϕ_2 DMA and if TxRQ is at "High" level, DMA transfer is performed at the following cycle. Therefore, a high-speed response (MAX. 1 byte/1 cycle) is feasible.

In burst mode, TxRQ should be also, in principle, set to "High" when I/O request is asserted, and reset to "Low" when $\overline{TxSTB}$ goes to "Low". If TxRQ is asserted as level input without being reset, DMA transfer is performed at all cycles of ϕ_2 DMA since TxRQ is always at "High" level at the falling edge of ϕ_2 DMA. Its example is shown in the second-byte and the third-byte transfer in Fig. 16.

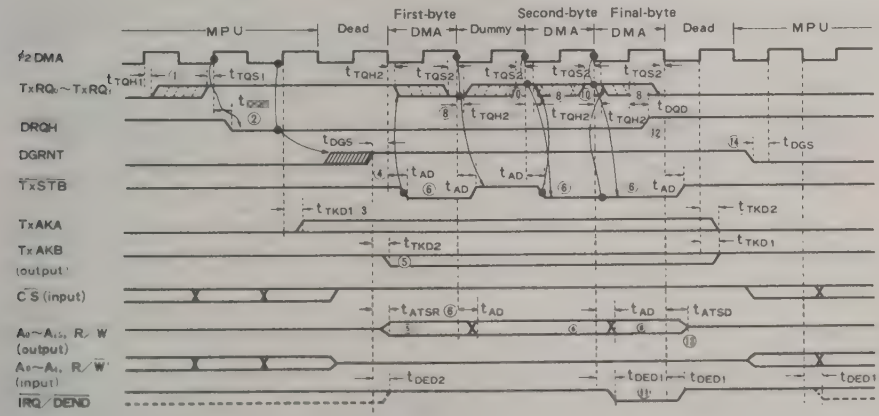


Figure 16 HALT Burst Mode

TSC Cycle Steal Mode

In the above-mentioned modes, DMA is performed by using the HALT function of the MPU. In TSC cycle steal mode, DMA is performed by using the TSC function of the MPU.

Its timing chart and flow chart are shown in Fig. 17 and Fig. 13 respectively.

Basic operation of the DMAC is the same as that in HALT cycle steal mode, but the detailed timing is different. The difference is explained in the following.

(1) $\overline{DRQT}$ is used for DMA transfer request instead of $\overline{DRQH}$.
(2) $\overline{DRQT}$ is sent to the external clock control circuit to extend clock E (ϕ_2) of MPU.
(3) To DGRNT, the external clock control circuit inputs response signals.

In TSC mode, there isn't a burst mode. Because the MPU clock cannot be extended for a long time because MPU performs dynamic operation. When TSC mode is specified, $\overline{DRQT}$ returns to "High" and the MPU gets into the instruction cycle everytime 1-byte transfer has finished.

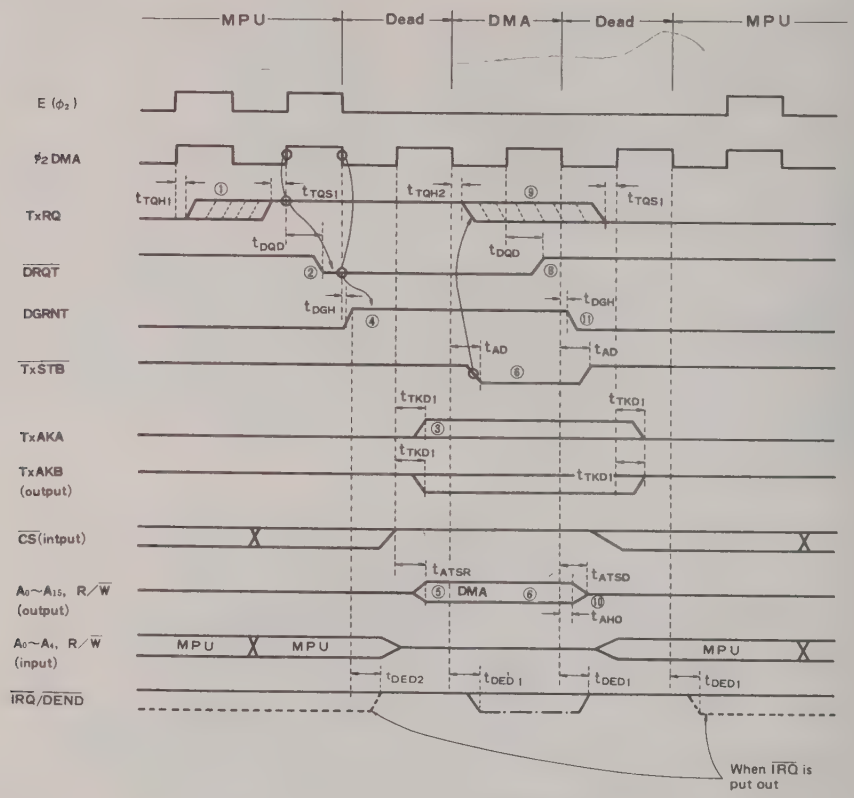


Figure 17 TSC Cycle Steal Mode

- **Priority Control**

Basic priority Control

There are two kinds of the DMAC priority control function. One is to mask TxRQ on each channel by TxRQ Enable bit of PCR. The other is priority-order-determining-circuit which the DMAC has as a hardware.

Moreover, the priority-order-determining-circuit has two operation modes (the rotate mode and the normal mode).

Structure of the priority control circuit is shown in Fig. 18. As shown in Fig. 18, TxRQ of the channel whose TxRQ Enable bit is at "1" level becomes an input of the priority-order-determining-circuit. Then it is checked whether TxRQ is at "High" level or not.

(Note) In this case, ZERO flag needs to be at "1" level. ZERO flag will be described later.

If one of $TxRQ_0 \sim TxRQ_3$ is at "High" level, its channel is selected, being given a first priority. Then it is latched by an executing-channel-number-latch-circuit to perform DMA transfer. Once an executing channel is determined and latched, it is unchanged until its DMA transfer has been completed. That is, the channel number strobe signal of DMAC doesn't go to "1" and the contents of the channel-number-latch-circuit are unchanged. In the cycle steal mode, the channel is fixed until 1-byte transfer has completed. In the burst mode, it is fixed until BCR becomes "0".

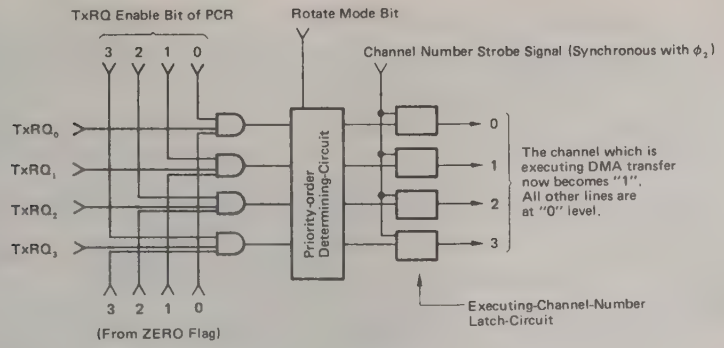


Figure 18 Structure of Priority Control Circuit

Therefore, once a long-period DMA transfer of a channel is performed in the burst mode, other channels need to wait until it has completed even if they have higher priority than the channel. Take much care to this point in designing response time to TxRQ of DMA channel.

(Note) As explained above, TxRQ input is latched internally. So once it is accepted and latched, the channel number cannot be changed even though it returns to "Low". But as explained in HALT Cycle Steal Mode, DMA transfer is not performed unless TxRQ rises to "High" again.

Strobe timing of executing-channel-number-latch-circuit which allow modification or decision of executing channel is shown in Fig. 19.

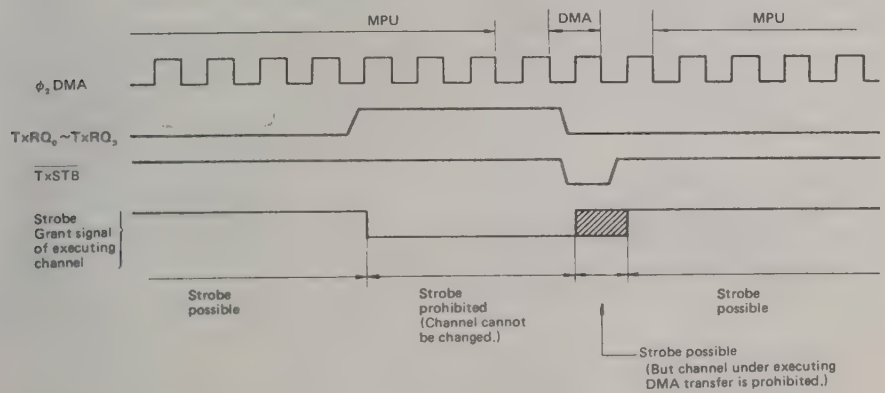


Figure 19 Strobe Timing of Executing-Channel-Number-Latch-Circuit (the cycle steal mode)

But, as shown in Fig. 19, only the channel under executing DMA transfer is prohibited to accept TxRQ during DMA transfer operation, in order that one more byte transfer may not be performed when the reset timing of TxRQ is delayed. Strobe timing in the burst mode is shown in Fig. 20.

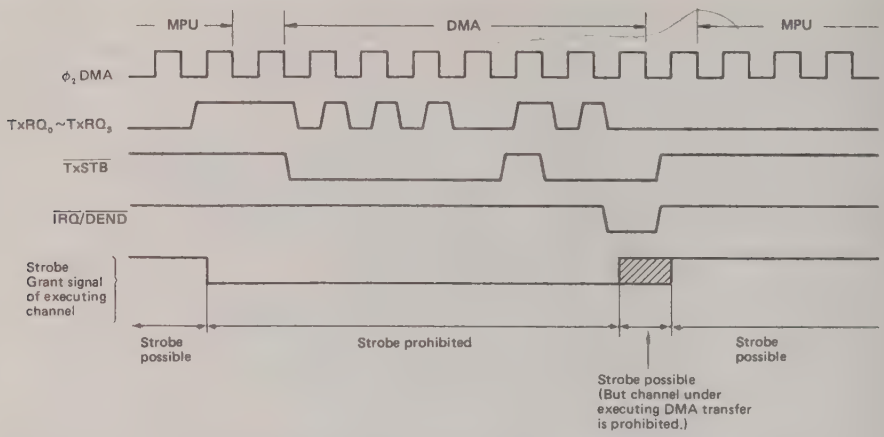


Figure 20 Strobe Timing of Executing-Channel-Number-Latch-Circuit (the burst mode)

Rotate Mode

There are two operation modes in priority-order-determining circuit. These are Normal Mode and Rotate Mode. In the normal mode, the order of priority is fixed at numerical order. (Channel 0 is given a first priority and then is followed by Channel 1 → 2 → 3.) In the rotate mode, the channel next to the channel with which DMA was executed in the last sequence, is given a first priority and the channel in the last sequence is given a last priority. But immediately after it gets into the reset state, the order of priority is the following: Channel 0 → 1 → 2 → 3.

An example of the rotate mode is shown in Fig. 21.

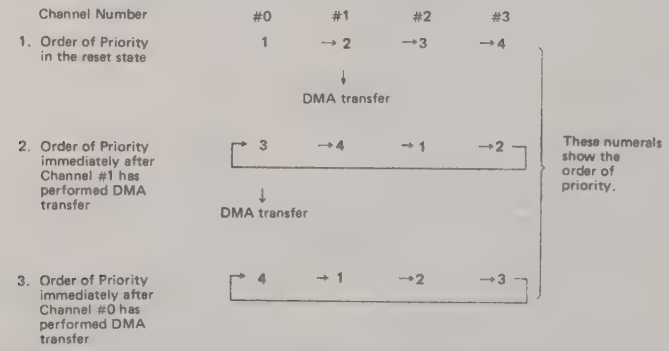
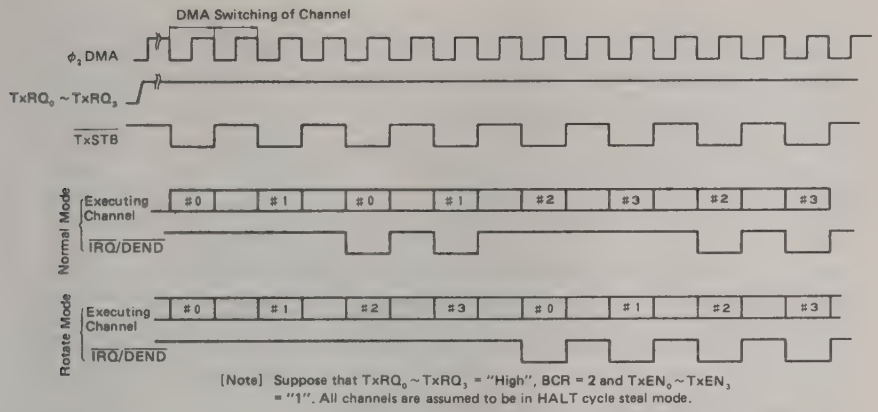


Figure 21 Example of Operation in the Rotate Mode

Next, Fig. 22 shows an example of the difference between the operation in the rotate mode and that in the normal mode. In this example, TxRQ of all channels is always at "High" level. Moreover, BCR=2 and TxEN=1 are assumed. As a transfer mode, HALT cycle steal mode is used.



[Note] Suppose that TxRQ$_0$ ~ TxRQ$_3$ = "High", BCR = 2 and TxEN$_0$ ~ TxEN$_3$ = "1". All channels are assumed to be in HALT cycle steal mode.

Figure 22 Difference between the operation in the rotate mode and that in the normal mode

The reason why the order of priority is not #0 → #0 → #1 → #1 → ——— in the normal mode is that during DMA transfer operation, TxRQ of an executing channel is prohibited from being accepted.

DMA Operation Timing with priority control

When more than 2 channels perform DMA transfer in parallel, the abovementioned priority-order-determining-circuit is used to determine the priority. The channel with lower priority waits until the channel with higher priority completes the transfer. Then it gets into DMA transfer operation. In this case, The following combinations of transfer modes are conceivable.

(1) From HALT mode to HALT mode (Fig. 23)
(2) From TSC mode to TSC mode (Fig. 24)
(3) From HALT mode to TSC mode
(4) From TSC mode to HALT mode } (Fig. 25)

In changing from HALT mode to HALT mode, only one dead cycle is intervened. That is, even in the cycle steal mode, DMA transfer of the next channel is performed without returning the bus control to the MPU ($\overline{DRQH}$ remains "Low").

In changing from TSC mode to TSC mode, DMA transfer of the next channel is performed, after returning the bus control to MPU for one cycle.

In the case of HALT → HALT, it doesn't return the bus control to MPU in order not to increase the response time of DMA transfer and dead cycles of the system.

On the other hand, in the case of TSC → TSC mode, same mean cannot be applicable because MPU clock cannot remain stopped for a long time as in the case of HALT mode.

Both in the case of HALT → TSC mode and in the case of TSC → HALT mode, DMA operation timing is based on the same idea as the above two kinds of mode change. (In detail, see Fig. 25).

The timing in the case where the next byte is transfered without changing the channel is shown in Fig. 26. This is the case of HALT → HALT mode. In this case, the bus control returns to MPU, before the next byte is transfered. In the case of TSC → TSC mode, its timing is almost the same as than in Fig. 24, that is, after 1-byte transfer has completed, MPU executes the Instruction Cycle for one clock and then DMAC executes 1-byte transfer again.

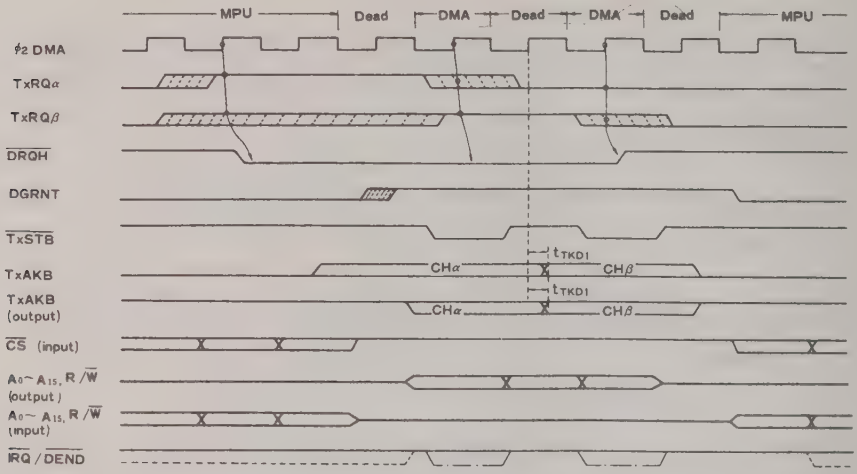


Figure 23 Channel Change (HALT Mode → HALT Mode)

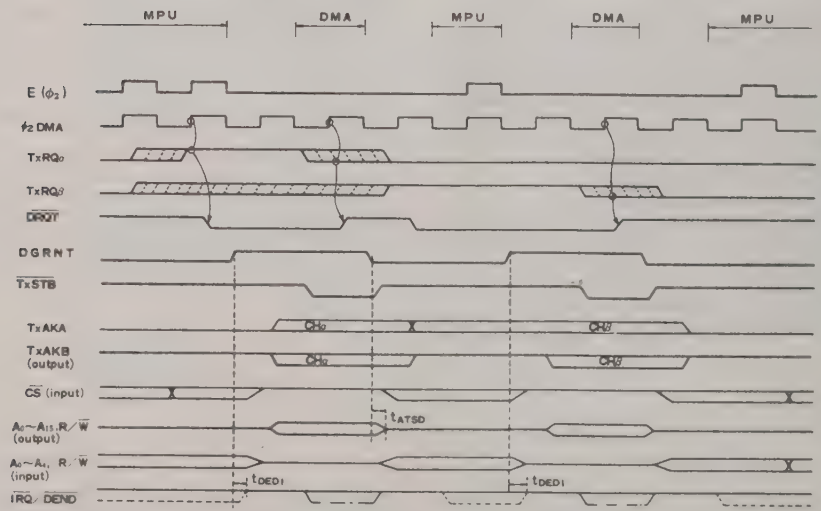


Figure 24 Channel Change (TSC Mode → TSC Mode)

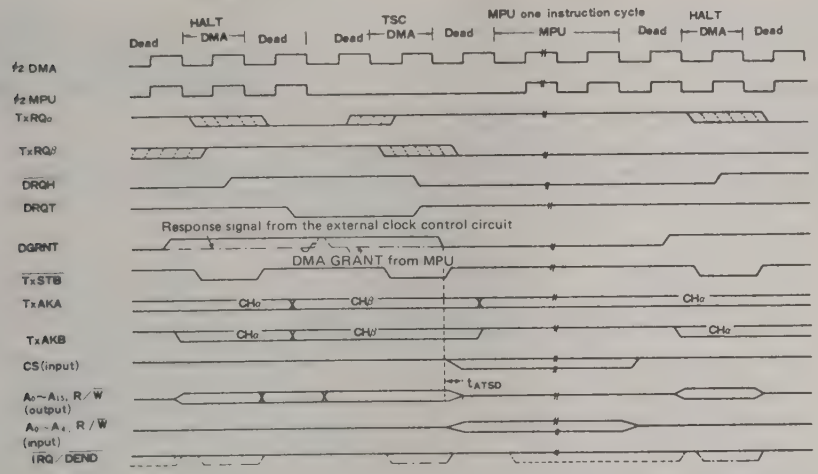


Figure 25 Channel Change (HALT Mode → TSC Mode → HALT Mode)

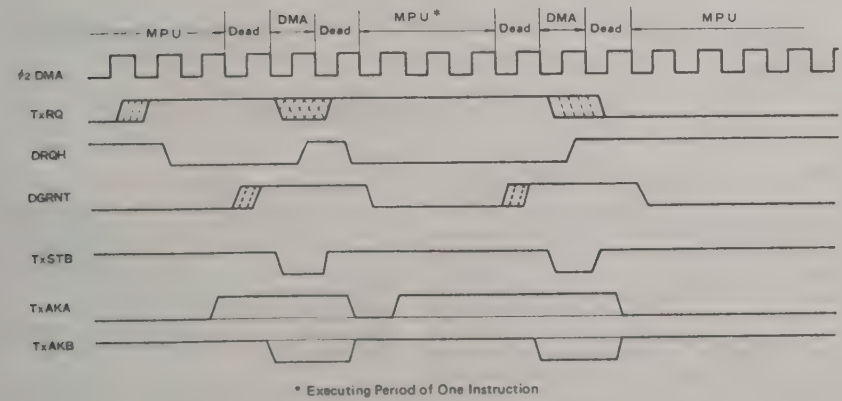


Figure 26 Successive 2-byte Transfer of One Channel (HALT Cycle Steal Mode)
HALT → HALT (by one channel)

- **Status Flag**

DMAC has BUSY Flag, DEND Flag and ZERO Flag on each channel. The former two of these flags can be read out by MPU, but ZERO Flag cannot be read out. Set and reset timing of each flag are shown in Fig. 27.

BUSY/READY Flag

This flag is set to "1" when it accepts the first-byte TxRQ of its corresponding channel. After 1-block transfer has completed and BCR becomes "0", it is reset to "0". Therefore, while this flag is "1", that is, its corresponding channel is being used, the next block transfer cannot be performed.

Also this flag is cleared when corresponding TxRQ Enable Bit in the PCR becomes "0".

DEND Flag

This is the interrupt flag to indicate the end of DMA transfer of its corresponding channel. After 1-block transfer has completed and BCR becomes "0", this flag is set to "1". This flag is reset to "0" immediately after the Channel Control Register having this flag is read out.

ZERO Flag

This is the internal flag to indicate whether the data stored in the BCR is "0" or not (It cannot be read out).

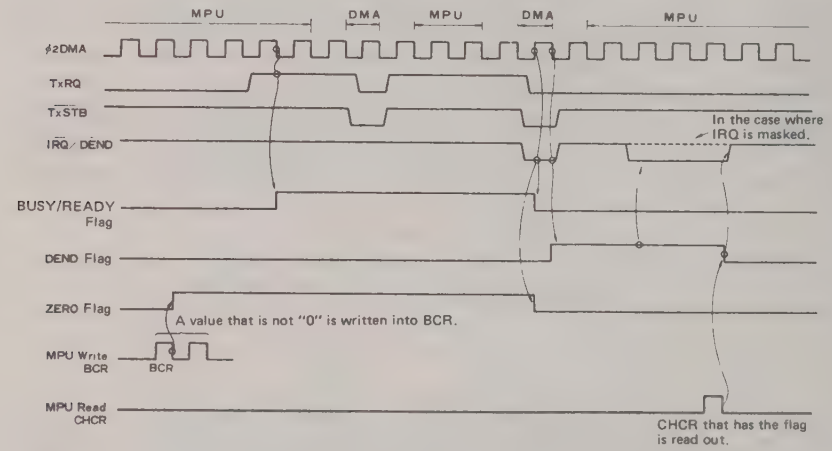


Figure 27 Timing of Status Flag (Suppose that BCR is 2 in the initial state)

When BCR is "0", ZERO Flag is "0". When BCR is not "0", it is "1".

In the reset state, this flag is "0". If data that is not "0" is written into BCR, this flag is set to "1". When BCR becomes "0" after 1-block data transfer has completed, or MPU writes "0" into BCR, this flag is reset to "0".

The function of ZERO Flag is to prohibit accepting TxRQ of its corresponding channel while this flag is "0" (that is, BCR is "0") (See Fig. 18). While ZERO Flag is "0", TxRQ is not accepted even if TxEN is "1". This function avoids an false operation even if "High" input is provided to TxRQ before the initialization of the register.

When $\overline{RES}$ pin goes to "Low", this flag becomes "0", but the number in BCR is not reset to "0". Therefore, the state of this flag and BCR are not the same. In this case new data should be written into BCR (Then ZERO Flag becomes "1").

- **DMA End Control**

Function of $\overline{IRQ}$/$\overline{DEND}$ Pin

DMAC has $\overline{IRQ}$ output and $\overline{DEND}$ output to perform DMA End Control. These are multiplexed outputs to $\overline{IRQ}$/$\overline{DEND}$ pin.

The function of $\overline{DEND}$ output is to inform I/O controller of the end of 1-block transfer. After 1-block transfer has been completed and BCR becomes "0", $\overline{DEND}$ output provides "Low" pulse whose cycle is one clock, being synchronous with the final 1-byte data transfer. 4 channels have only one $\overline{DEND}$ output in common, so each channel determines whether $\overline{DEND}$ output is its own output or not, combining with TxAK signal. When TxAK of the channel is "High" and $\overline{DEND}$ is "Low", it shows that the cycle is the last one of DMA (See Fig. 29 and 30).

The function of $\overline{IRQ}$ output is to inform MPU of the end of 1-block transfer by interrupting it. As shown in Fig. 28, $\overline{IRQ}$ output is logical AND-OR of the interrupt flag (DEND Flag) and IRQ Enable bit of each channel.

$\overline{IRQ}$ and $\overline{DEND}$ outputs are multiplexed. $\overline{IRQ}$/$\overline{DEND}$ pin is used as $\overline{DEND}$ output during DMAC cycle and $\overline{IRQ}$ output during MPU cycle. Moreover, DGRNT signal separates $\overline{DEND}$ and $\overline{IRQ}$ by its "High" or "Low". In detail, see Fig. 29 and Fig. 30.

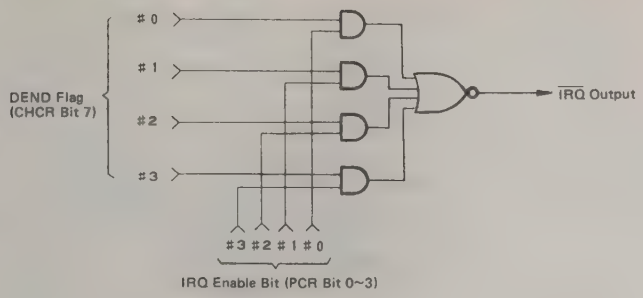


Figure 28 Logic of $\overline{IRQ}$ Output

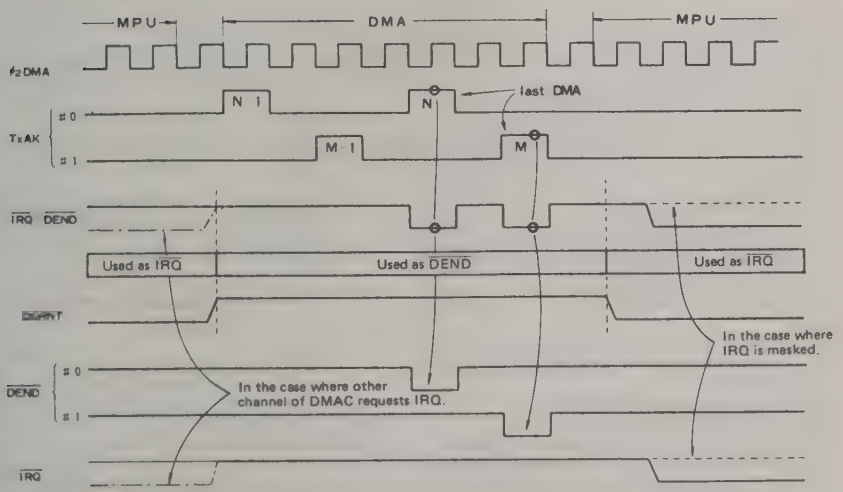


Figure 29 Timing of $\overline{IRQ}$/$\overline{DEND}$ Output

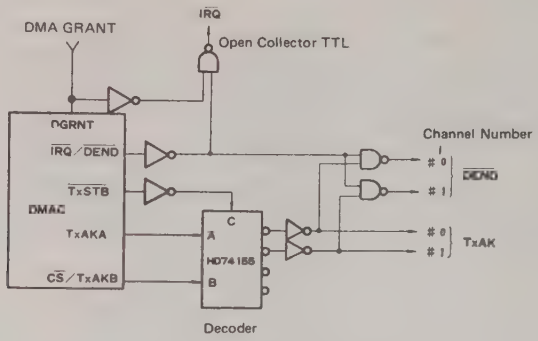


Figure 30 How to Use $\overline{IRQ}/\overline{DEND}$ Output Signal

Unusual DMA End

Following section describes how to terminate or change normal sequence of DMA transfer.

(1) When "0" is written into BCR
When "0" is written into BCR before it becomes "0", subsequent TxRQ are not accepted and this causes the termination of the DMA transfer since the internal ZERO Flag is reset to "0". In this case, note that $\overline{DEND}$ pulse is not provided.

(2) When "1" is written into BCR
When "1", instead of "0", is written into BCR, only the next TxRQ is accepted and 1-byte DMA transfer is performed. In this case, $\overline{DEND}$ pulse is provided, being synchronous with the last transfer.

(3) When another value is written into ADR & BCR during the transfer
When the data in ADR & BCR are changed during the transfer, the following transfer is performed according to the change of the data.

(4) When "0" is written into TxRQ Enable bit
When TxEN is reset to "0" during the transfer, this causes TxRQ comes not to be accepted and the transfer halts. But the state is different from that in the case (1), the number in BCR remains unchanged. Therefore, when TxEN is set to "1" again, the transfer is performed again.

(5) When $\overline{RES}$ pin is set to "Low"
When $\overline{RES}$ is provided during the transfer, the transfer stops.
Then all of the control registers and their internal flags are reset to "0". But the data in ADR & BCR are not reset.

(Supplement)

It is only in the cycle steal mode that DMAC registers such as BCR and ADR can be read or written during the transfer. In the burst mode, it is usually impossible (But special external circuits enable it).

- **Data Chain Function**

The data chain function of DMAC is to transfer the contents of ADR & BCR of Channel #3 to ADR & BCR of a specified channel automatically and renew the data of them after the channel has completed 1-block transfer.

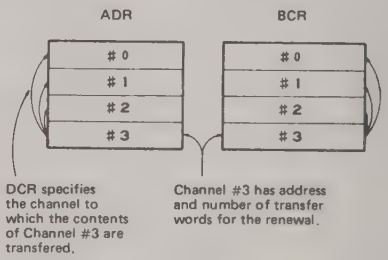


Figure 31 Data Chain Operation

Its detailed timing is shown in Fig. 32 and Fig. 33. As shown in these figures the contents of ADR & BCR of Channel #3 are transfered to the channel during the clock cycle next to the last one of 1-block transfer (which provides DEND pulse). Then $\overline{DRQH}$ or $\overline{DRQT}$ provides "Low" output for one more clock cycle than in the normal case. Therefore, MPU takes back the bus control again 1-clock later than in the normal case, that is, after the data renewal of the specified channel by the data chain from Channel #3.

In the TSC mode, the stretching period of clockϕ_1 is longer than in the normal case.

The contents of ADR & BCR of Channel #3 remain unchanged as long as new data are not written by MPU, even if the data chain is executed.

As for $\overline{DEND}$ output, DEND Flag and BUSY Flag in the case of data chain execution, they function in the same way as in the normal case. They provide $\overline{DEND}$ pulse everytime 1-block transfer has completed, and then DEND Flag is set to "1". Therefore, in the case where more than 3-block data chain is needed, DEND Flag is used for the execution. Its sequence is shown in Fig. 34. First, DEND Flag="1" that shows the end of the first-block data chain is read out. Next, the data of ADR & BCR for the third-block data chain need to be written into Channel #3, in parallel with the execution of the second-block data chain. (This data chain is feasible only in the cycle steal mode.)

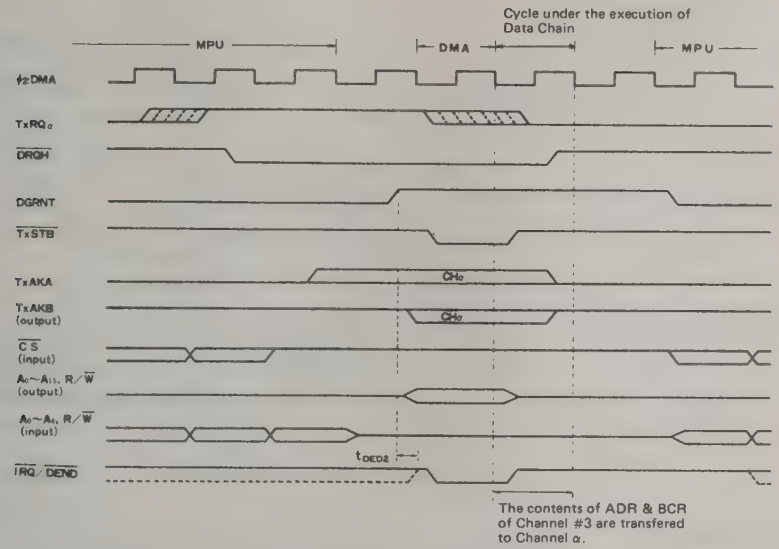


Figure 32 Data Chain Operation (HALT Mode)

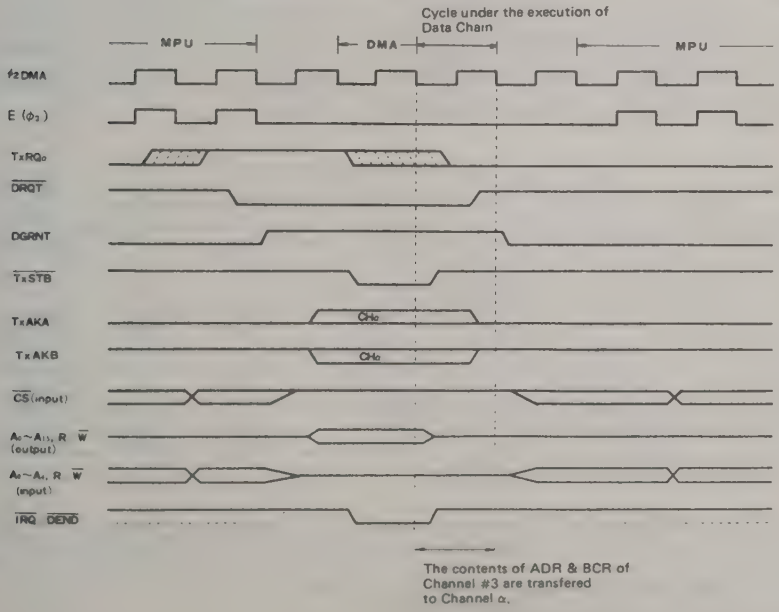


Figure 33 Data Chain Operation (TSC Mode)

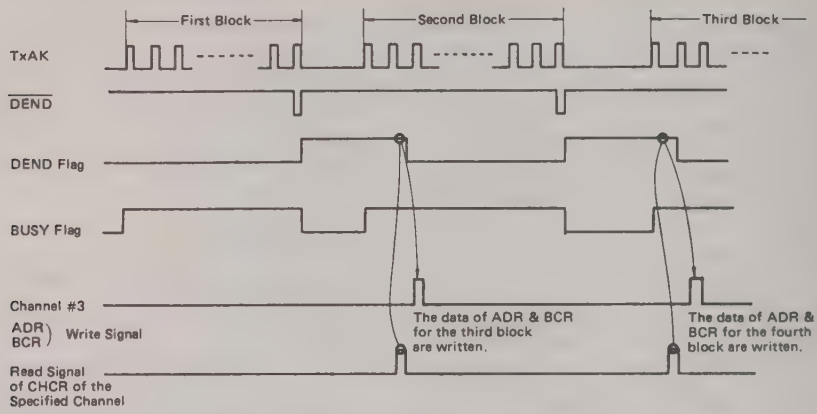


Figure 34 Sequence of More than 3-block Data Chain

■ DMAC PROGRAMMING

Preparation of a channel for a DMA transfer requires:

1) Load the starting address into the Address Register.
2) Load the number of bytes into the Byte Count Register.
3) Program the Channel Control Register for the transfer characteristics: direction (bit 0), mode (bits 1 and 2), and the address update (bit 3).

The channel is now configured. To enable the transfer request, set the appropriate enable bit (bits 0~3) of the Priority Control Register, as well as the Rotate Control bit.

If an interrupt on DMA End is desired, the enable bit (bits 0~3) of the Interrupt Control Register must be set.

If data chaining for the channel is necessary, it is programmed into the Data Chain Register and the appropriate data must be written into the Address and Byte Count Registers for channel #3.

Table 8 DMAC Programming Model

Register	Address (Hex)	Register Content							
		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Channel Control	1x*	DMA End Flag (DEND)	Busy/Ready Flag	Not Used	Not Used	Address Up/Down	TSC/ Halt	Burst/ Steal	Read/Write (R/W)
Priority Control	14	Rotate Control	Not Used	Not Used	Not Used	TxRQ Enable #3 (TxEN3)	TxRQ Enable #2 (TxEN2)	TxRQ Enable #1 (TxEN1)	TxRQ Enable #0 (TxEN0)
Interrupt Control	15	IRQ Flag	Not Used	Not Used	Not Used	IRQ Enable #3 (IE3)	IRQ Enable #2 (IE2)	IRQ Enable #1 (IE1)	IRQ Enable #0 (IE0)
Data Chain	16	Not Used	Not Used	Not Used	Not Used	Two/Four Channel Select (2/4)	Data Chain Channel Select B	Data Chain Channel Select A	Data Chain Enable

* The x represents the binary equivalent of the channel desired.

A comparison of the response times and maximum transfer rates is shown in Table 9. The data are shown for a system clock rate of 1 MHz.

The two 8-bit bytes that form the registers in Table 10 are placed in consecutive memory locations, making it very easy to use the MPU index register in programming them.

Fig. 38 shows an example of its minimum structure (1 channel, HALT mode, combination with FDC). Fig. 39 shows an example of its maximum structure. (but only one DMAC is used.)

Table 9 Maximum Transfer Speed & Response Time of the DMAC when $t_{cyc\phi}$ equals 1 μsec.

Mode		Maximum Transfer Speed (μsec/byte)	Response Time (μsec)	
			maximum	minimum
HALT Cycle Steal		(executing time of one instruction) + 3	(executing time of one instruction) $+3.5 - t_{TQH1}$	$3.5 + t_{TQS1}$
HALT Burst	first byte	1		
	since second byte		$2 - t_{TQH2}$	$1 + t_{TQS2}$
TSC Cycle Steal		4	$3.5 - t_{TQH1}$	$2.5 + t_{TQH1}$

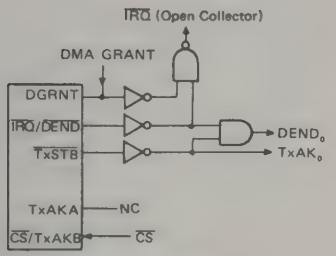


Figure 35 One Channel

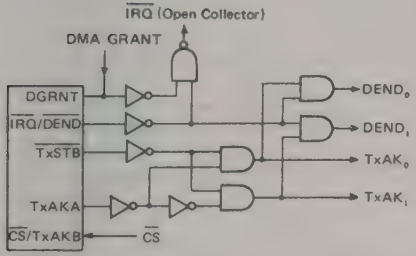


Figure 36 Two Channel

Table 10 Address and Byte Count Registers

Register	Channel	Address (Hex)
Address High	0	0
Address Low	0	1
Byte Count High	0	2
Byte Count Low	0	3
Address High	1	4
Address Low	1	5
Byte Count High	1	6
Byte Count Low	1	7
Address High	2	8
Address Low	2	9
Byte Count High	2	A
Byte Count Low	2	B
Address High	3	C
Address Low	3	D
Byte Count High	3	E
Byte Count Low	3	F

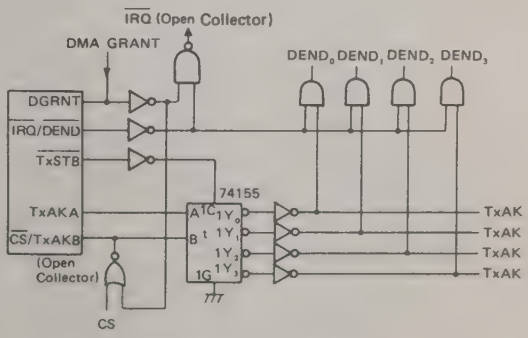


Figure 37 Four-Channel

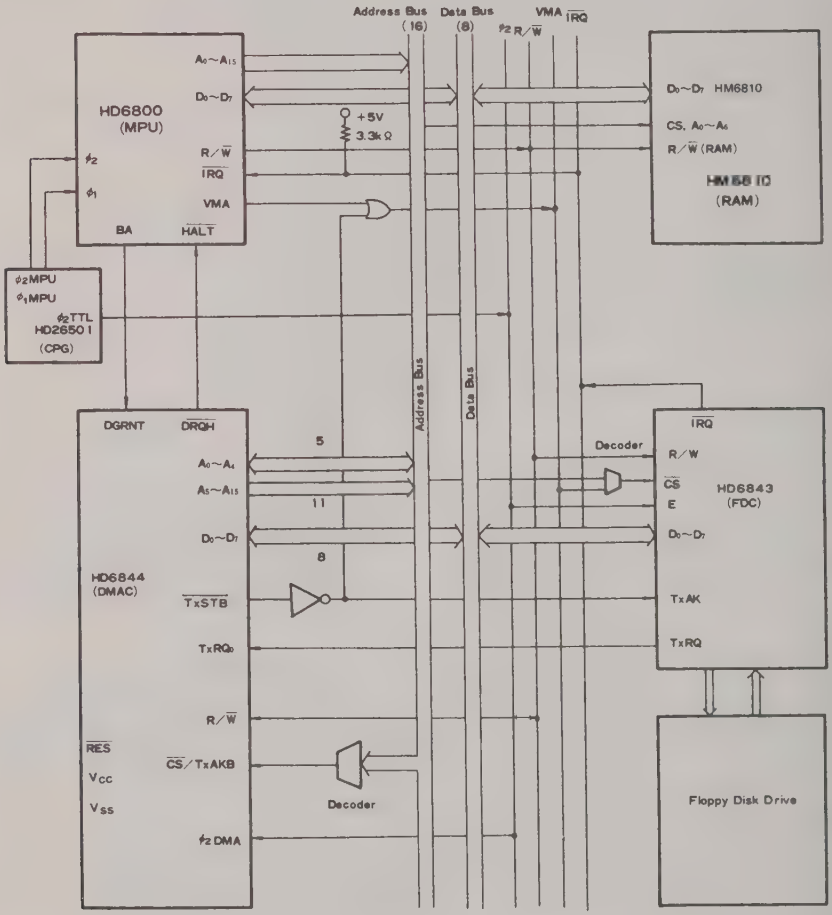


Figure 38 Example of DMA System Structure (1) (minimum)

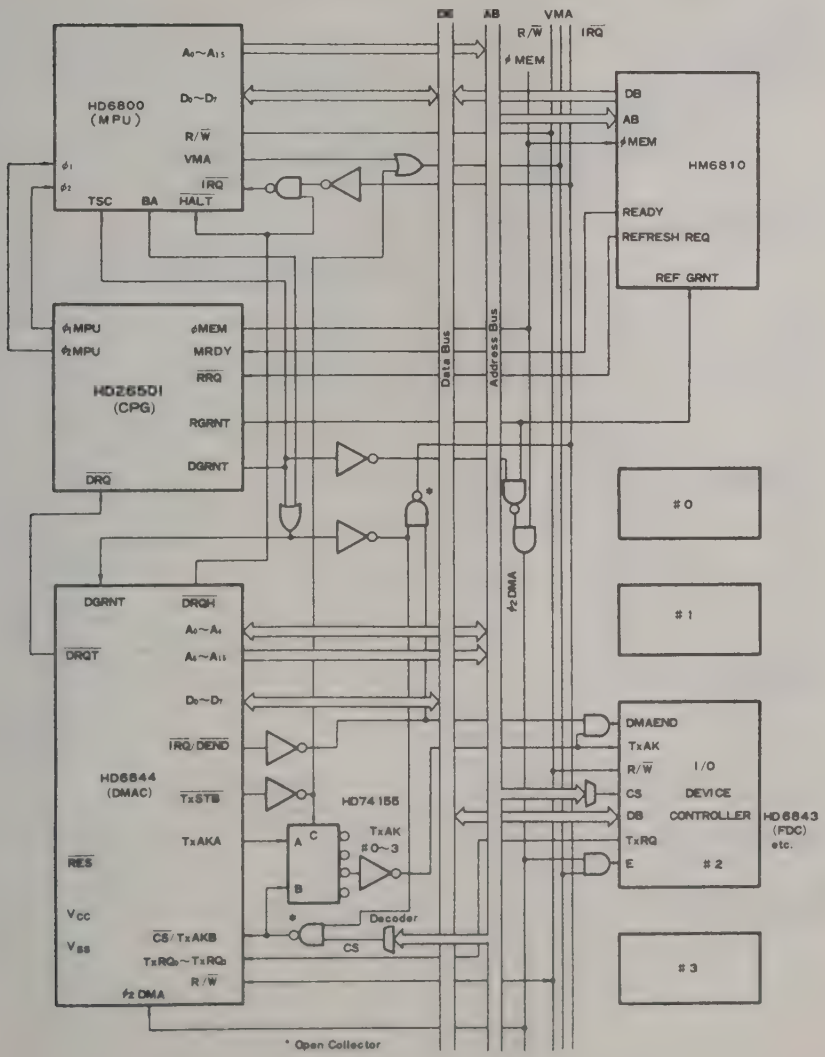
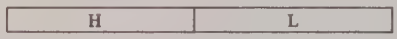


Figure 39 Example of DMA System Structure (2) (maximum)

■ APPENDIX

Contents of the DMAC Registers

(1) ADR0 ~ ADR3 (Address Register) (1 ADR on each channel)

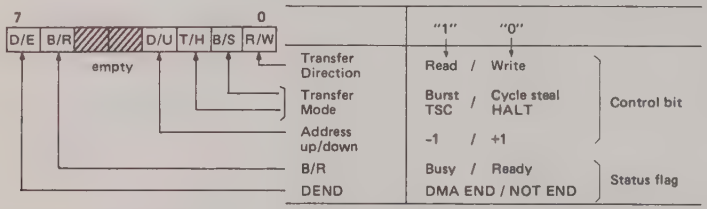


16 bit x 4

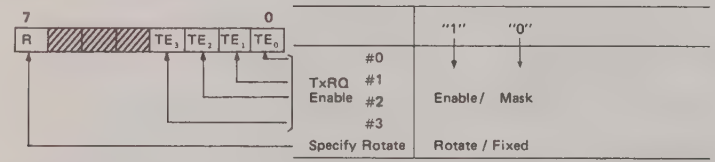
(2) BCR0 ~ BCR3 (Byte Count Register) (1 BCR on each channel)

16 bit x 4

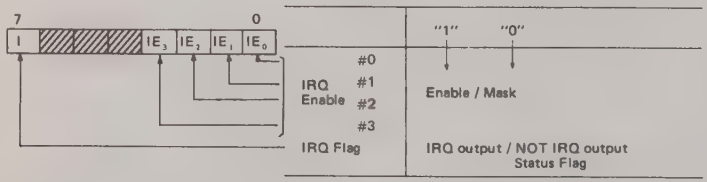
(3) CHCR0 ~ CHCR3 (Channel Control Register) (1 CHCR on each channel) (6 bit x 4)



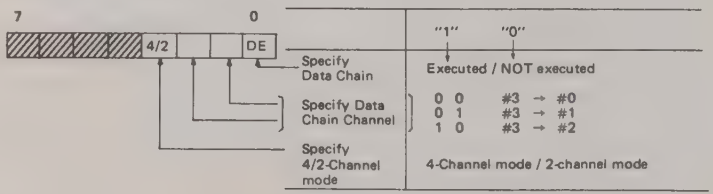
(4) PCR (Priority Control Register) (5 bit x 1)



(5) ICR (Interrupt Control Register) (5 bit x 1)



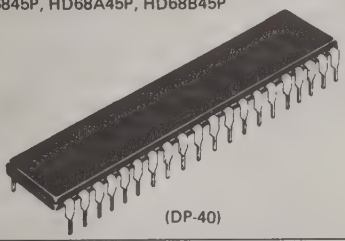
(6) DCR (Data Chain Control Register) (4 bit x1)



HD6845, HD68A45, HD68B45
CRTC (CRT Controller)

The CRTC is a LSI controller which is designed to provide an interface for microcomputers to raster scan type CRT displays. The CRTC belongs to the HMCS6800 LSI Family and has full compatibility with MPU in both data lines and control lines. Its primary function is to generate timing signal which is necessary for raster scan type CRT display according to the specification programmed by MPU. The CRTC is also designed as a programmable controller, so applicable to wide-range CRT display from small low-functioning character display up to raster type full graphic display as well as large high-functioning limited graphic display.

HD6845P, HD68A45P, HD68B45P

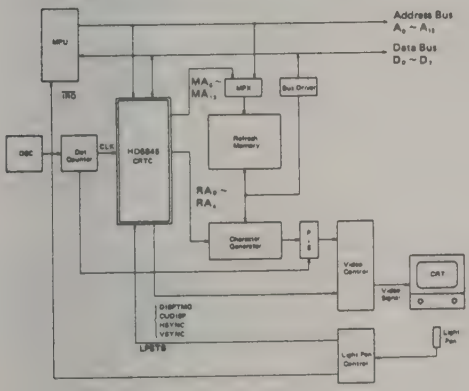


(DP-40)

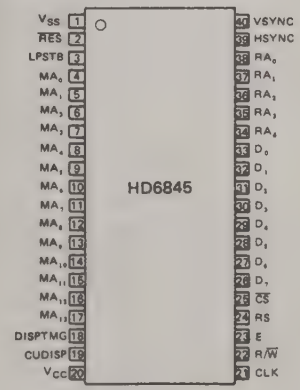
■ FEATURES

- Number of Displayed Characters on the Screen, Vertical Dot Format of One Character, Horizontal and Vertical Sync Signal, Display Timing Signal are Programmable
- Line Buffer-less Refreshing
- 14-bit Refresh Memory Address Output (16k Words max. Access)
- Programmable Interlace/Non-interlace Scan Mode
- Built-in Cursor Control Function
- Programmable Cursor Height and its Blink
- Built-in Light Pen Detection Function
- Paging and Scrolling Capability
- TTL Compatible
- Single +5V Power Supply
- Compatible with MC6845, MC68A45, MC68B45

■ SYSTEM BLOCK DIAGRAM



■ PIN ARRANGEMENT



(Top View)

■ ORDERING INFORMATION

CRTC	Bus Timing	CRT Display Timing
HD6845	1.0 MHz	3.0 MHz max.
HD68A45	1.5 MHz	
HD68B45	2.0 MHz	

■ ABSOLUTE MAXIMUM RATINGS

Item	Symbol	Value	Unit
Supply Voltage	V_{CC} *	−0.3 ~ +7.0	V
Input Voltage	V_{in} *	−0.3 ~ +7.0	V
Operating Temperature	T_{opr}	− 20 ~ + 75	°C
Storage Temperature	T_{stg}	− 55 ~ +150	°C

* With respect to V_{SS} (SYSTEM GND)

[NOTE] Permanent LSI damage may occur if maximum ratings are exceeded. Normal operation should be under recommended operating conditions. If these conditions are exceeded, it could affect reliability of LSI.

■ RECOMMENDED OPERATING CONDITIONS

Item	Symbol	min	typ	max	Unit
Supply Voltage	V_{CC} *	4.75	5.0	5.25	V
Input Voltage	V_{IL} *	−0.3	–	0.8	V
	V_{IH} *	2.0	–	V_{CC}	V
Operating Temperature	T_{opr}	− 20	25	75	°C

* With respect to V_{SS} (SYSTEM GND)

■ ELECTRICAL CHARACTERISTICS

• DC CHARACTERISTICS (V_{CC} = 5V ± 5%, V_{SS} = 0V, Ta = −20~+75°C, unless otherwise noted.)

Item	Symbol	Test Condition		min	typ*	max	Unit
Input "High" Voltage	V_{IH}			2.0	–	V_{CC}	V
Input "Low" Voltage	V_{IL}			−0.3	–	0.8	V
Input Leakage Current	I_{in}	V_{in} = 0 ~ 5.25V (Except D_0~D_7)		−2.5	–	2.5	μA
Three-State Input Current (off-state)	I_{TSI}	V_{in} = 0.4 ~ 2.4V V_{CC} = 5.25V (D_0~ D_7)		− 10	–	10	μA
Output "High" Voltage	V_{OH}	I_{LOAD} = −205 μA (D_0~ D_7)		2.4	–	–	V
		I_{LOAD} = −100 μA (Other Outputs)					
Output "Low" Voltage	V_{OL}	I_{LOAD} = 1.6 mA		–	–	0.4	V
Input Capacitance	C_{in}	V_{in} = 0 Ta = 25°C f = 1.0 MHz	D_0 ~ D_7	–	–	12.5	pF
			Other Inputs	–	–	10.0	pF
Output Capacitance	C_{out}	V_{in} = 0V, Ta = 25°C, f = 1.0 MHz		–	–	10.0	pF
Power Dissipation	P_D			–	600	1000	mW

* Ta = 25°C, V_{CC} = 5.0V

- **AC CHARACTERISTICS (V_{CC} = 5V ±5%, V_{SS} = 0V, Ta = −20~+75°C, unless otherwise noted.)**

1. TIMING OF CRTC SIGNAL

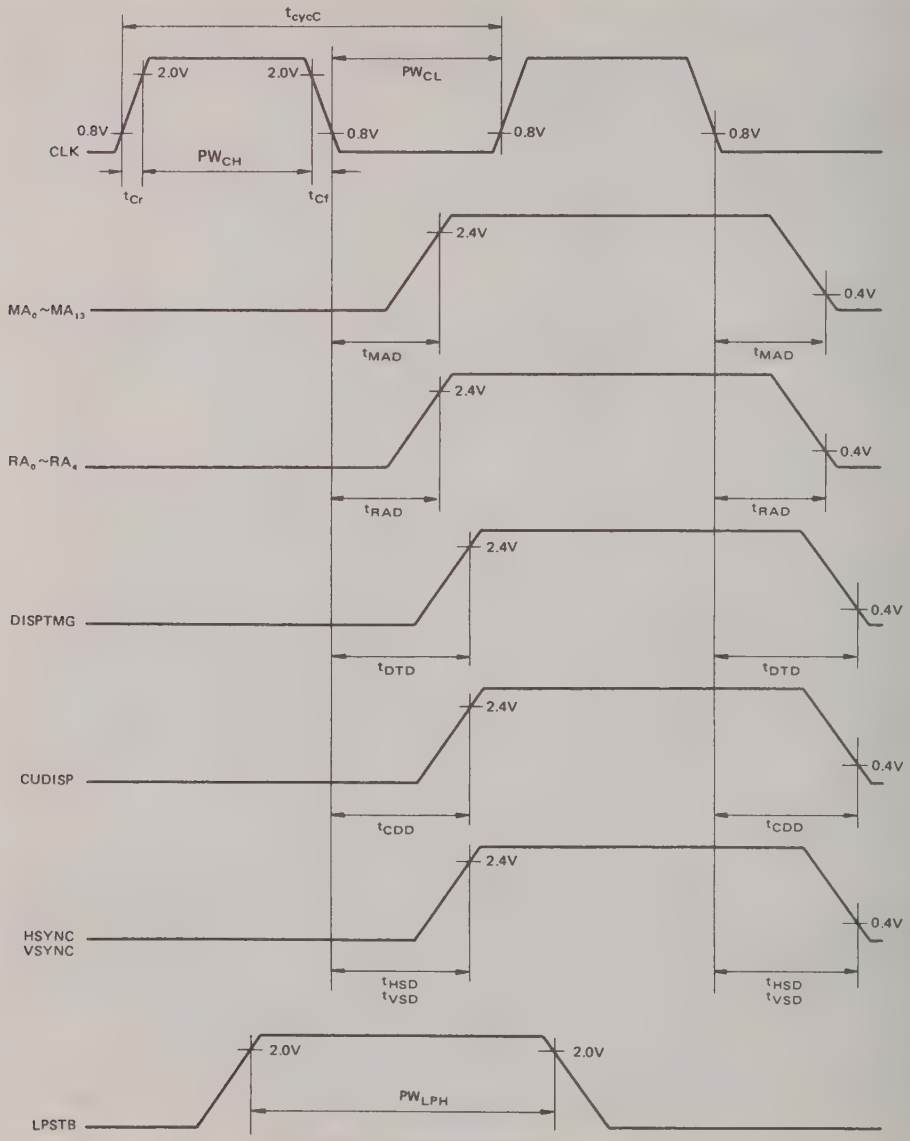
Item	Symbol	Test Condition	min	typ	max	Unit
Clock Cycle Time	t_{cycC}	Fig. 1	330	–	–	ns
Clock "High" Pulse Width	PW_{CH}		150	–	–	ns
Clock "Low" Pulse Width	PW_{CL}		150	–	–	ns
Rise and Fall Time for Clock Input	t_{Cr}, t_{Cf}		–	–	15	ns
Memory Address Delay Time	t_{MAD}		–	–	160	ns
Raster Address Delay Time	t_{RAD}		–	–	160	ns
DISPTMG Delay Time	t_{DTD}		–	–	250	ns
CUDISP Delay Time	t_{CDD}		–	–	250	ns
Horizontal Sync Delay Time	t_{HSD}		–	–	250	ns
Vertical Sync Delay Time	t_{VSD}		–	–	250	ns
Light Pen Strobe Pulse Width	PW_{LPH}		80	–	–	ns
Light Pen Strobe	t_{LPD1}	Fig. 2	–	–	80	ns
Uncertain Time of Acceptance	t_{LPD2}		–	–	10	ns

2. MPU READ TIMING

Item	Symbol	Test Condition	HD6845			HD68A45			HD68B45			Unit
			min	typ	max	min	typ	max	min	typ	max	
Enable Cycle Time	t_{cycE}	Fig. 3	1.0	–	–	0.666	–	–	0.5	–	–	μs
Enable "High" Pulse Width	PW_{EH}		0.45	–	–	0.28	–	–	0.22	–	–	μs
Enable "Low" Pulse Width	PW_{EL}		0.40	–	–	0.28	–	–	0.21	–	–	μs
Enable Rise and Fall Time	t_{Er}, t_{Ef}		–	–	25	–	–	25	–	–	25	ns
Address Set Up Time	t_{AS}		140	–	–	140	–	–	70	–	–	ns
Data Delay Time	t_{DDR}		–	–	320	–	–	220	–	–	180	ns
Data Hold Time	t_H		10	–	–	10	–	–	10	–	–	ns
Address Hold Time	t_{AH}		10	–	–	10	–	–	10	–	–	ns
Data Access Time	t_{ACC}		–	–	460	–	–	360	–	–	250	ns

3. MPU WRITE TIMING

Item	Symbol	Test Condition	HD6845			HD68A45			HD68B45			Unit
			min	typ	max	min	typ	max	min	typ	max	
Enable Cycle Time	t_{cycE}	Fig. 4	1.0	–	–	0.666	–	–	0.5	–	–	μs
Enable "High" Pulse Width	PW_{EH}		0.45	–	–	0.28	–	–	0.22	–	–	μs
Enable "Low" Pulse Width	PW_{EL}		0.40	–	–	0.28	–	–	0.21	–	–	μs
Enable Rise and Fall Time	t_{Er}, t_{Ef}		–	–	25	–	–	25	–	–	25	ns
Address Set Up Time	t_{AS}		140	–	–	140	–	–	70	–	–	ns
Data Set Up Time	t_{DSW}		195	–	–	80	–	–	60	–	–	ns
Data Hold Time	t_H		10	–	–	10	–	–	10	–	–	ns
Address Hold Time	t_{AH}		10	–	–	10	–	–	10	–	–	ns



This Figure shows the relation in time between CLK signal and each output signals. Output sequence is shown in Figs. 10~15.

Figure 1 Time Chart of the CRTC

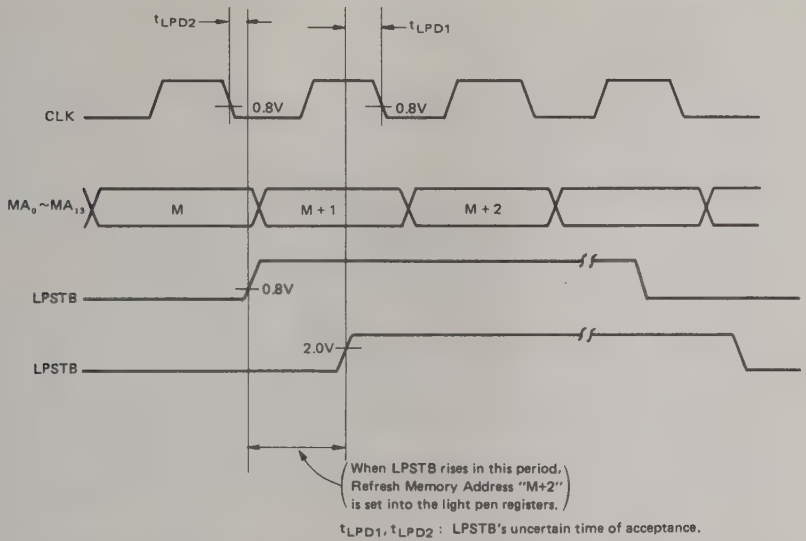


Figure 2 LPSTB Input Timing & Refresh Memory Address that is set into the light pen registers.

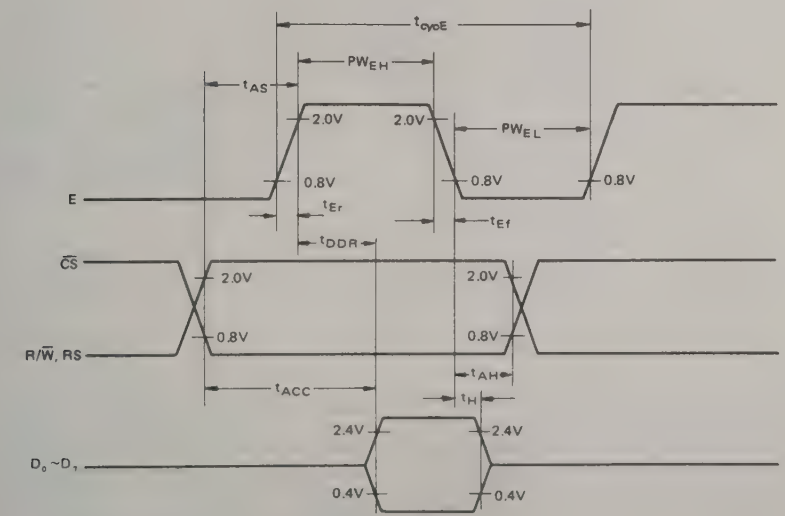


Figure 3 Read Sequence

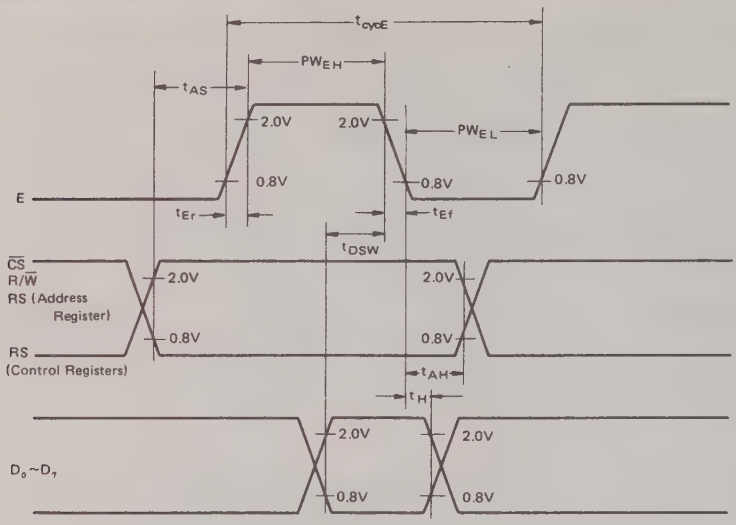
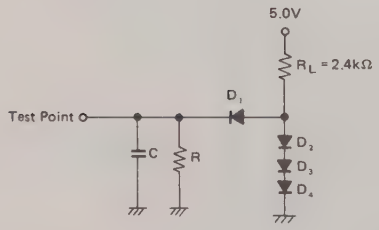


Figure 4 Write Sequence



C = 130pF (D_0 ~ D_7)
= 30pF (Output signals except D_0 ~ D_7)
R = 11kΩ (D_0 ~ D_7)
= 24kΩ (Output signals except D_0 ~ D_7)
D_1 ~ D_4 are 1S2074Ⓗ or equivalent.

Figure 5 Test Loads

■ SYSTEM DESCRIPTION

The CRTC is a LSI which is connected with MPU and CRT display device to control CRT display. The CRTC consists of internal register group, horizontal and vertical timing circuits, linear address generator, cursor control circuit, and light pen detection circuit. Horizontal and vertical timing circuit generate RA_0~RA_4, DISPTMG, HSYNC, and VSYNC. RA_0~RA_4 are raster address signals and used as input signals for Character Generator. DISPTMG, HSYNC, and VSYNC signals are received by video control circuit. This horizontal and vertical timing circuit consists of internal counter and comparator circuit.

Linear address generator generates refresh memory address MA_0 ~MA_{13} to be used for refreshing the screen. By these address signals, refresh memory is accessed periodically. As 14 refresh memory address signals are prepared, 16k words max are accessible. Moreover, the use of start address register enables paging and scrolling. Light pen detection circuit detects light pen position on the screen. When light pen strobe signal is received, light pen register memorizes linear address generated by linear address generator in order to memorize where light pen is on the screen. Cursor control circuit controls the position of cursor, its height, and its blink.

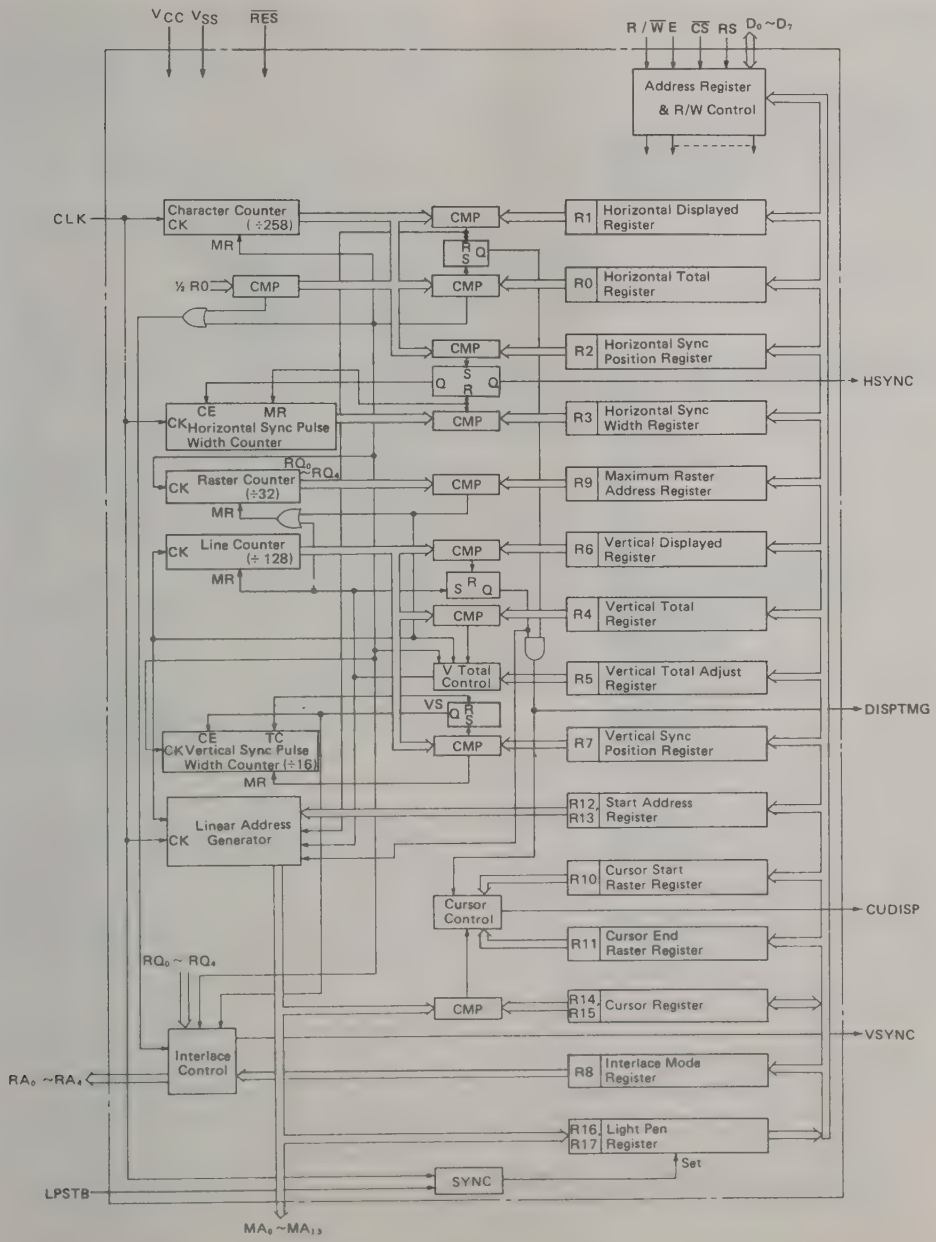


Figure 6 Internal Block Diagram of the CRTC

■ FUNCTION OF SIGNAL LINE

The CRTC provides 13 interface signals to MPU and 25 interface signals to CRT display.

● Interface Signals to MPU

Bi-directional Data Bus (D_0~D_7)

Bi-directional data bus(D_0~D_7) are used for data transfer between the CRTC and MPU. The data bus outputs are 3-state buffers and remain in the high-impedance state except when MPU performs a CRTC read operation.

Read/Write (R/$\overline{W}$)

Read/Write signal (R/$\overline{W}$) controls the direction of data transfer between the CRTC and MPU. When R/$\overline{W}$ is at "High" level, data of CRTC is transfered to MPU. When R/$\overline{W}$ is at "Low" level, data of MPU is transfered to CRTC.

Chip Select ($\overline{CS}$)

Chip Select signal ($\overline{CS}$) is used to address the CRTC. When $\overline{CS}$ is at "Low" level, it enables Read/Write operation to CRTC internal registers. Normally this signal is derived from decoded address signal of MPU under the condition that VMA of MPU is at "High" level.

Register Select (RS)

Register Select signal (RS) is used to select the address register and 18 control registers of the CRTC. When RS is at "Low" level, the address register is selected and when RS is at "High" level, control registers are selected. This signal is normally a derivative of the lowest bit (A0) of MPU address bus.

Enable(E)

Enable signal (E) is used as strobe signal in MPU Read/Write operation with the CRTC internal registers. This signal is normally a derivative of the HMCS6800 System ϕ_2 clock.

Reset ($\overline{RES}$)

Reset signal ($\overline{RES}$) is an input signal used to reset the CRTC. When $\overline{RES}$ is at "Low" level, it forces the CRTC into the following status.

1) All the counters in the CRTC are cleared and the device stops the display operation*.
2) All the outputs go down to "Low" level. (D_0~D_7 are not affected.)
3) Control registers in the CRTC are not affected and remain unchanged.

The CRTC internal registers can be accessed by MPU even in reset status.

This signal is different from other HMCS6800 family LSIs in the following functions and has restrictions for usage.

1) $\overline{RES}$ has capability of reset function only when LPSTB is at "Low" level.
2) The CRTC starts the display operation immediately after $\overline{RES}$ goes "High" level.
3) After $\overline{RES}$ has gone down to "Low" level, MA_0 ~ MA_{13} and RA_0 ~ RA_4, synchronizing with CLK "Low" level, go down to "Low" level.
(At least 1 cycle CLK is necessary to reset.)

*In the case "0" is set to Horizontal Sync Position Register (R2), CLK signal is output from HSYNC pin.

● Interface Signals to CRT Display Device

Character Clock (CLK)

CLK is a standard clock input signal which defines character timing for the CRTC display operation. CLK is normally derived from the external high-speed dot timing logic.

Horizontal Sync (HSYNC)

HSYNC is an active "High" level signal which provides horizontal synchronization for display device.

Vertical Sync (VSYNC)

VSYNC is an active "High" level signal which provides vertical synchronization for display device.

The pulse width is fixed at 16H.

Display Timing (DISPTMG)

DISPTMG is an active "High" level signal which defines the display period in horizontal and vertical raster scanning. It is necessary to enable video signal only when DISPTMG is at "High" level.

Refresh Memory Address (MA_0~MA_{13})

MA_0~MA_{13} are refresh memory address signals which are used to access to refresh memory in order to refresh the CRT screen periodically. These outputs enables 16k words max. refresh memory access. So, for instance, these are applicable up to 2000 characters/screen and 8-page system.

Raster Address (RA_0~RA_4)

RA_0~RA_4 are raster address signals which are used to select the raster of the character generator or graphic pattern generator etc.

Cursor Display (CUDISP)

CUDISP is an active "High" level video signal which is used to display the cursor on the CRT screen. This output is inhibited while DISPTMG is at "Low" level. Normally this output is mixed with video signal and provided to the CRT display device.

Light Pen Strobe (LPSTB)

LPSTB is an active "High" level input signal which accepts strobe pulse detected by the light pen and control circuit. When this signal is activated, the refresh memory address (MA_0~MA_{13}) which are shown in Fig. 2 are stored in the 14-bit light pen register. The stored refresh memory address need to be corrected in software, taking the delay time of the display device, light pen, and light pen control circuits into account.

■ REGISTER DESCRIPTION

Table 1 Internal Registers Assignment

$\overline{CS}$	RS	Address Register 4 3 2 1 0	Register #	Register Name	Program Unit	READ	WRITE	Data Bit 7	6	5	4	3	2	1	0
1	×	× × × × ×			–	–	–								
0	0	× × × × ×	AR	Address Register	–	×	○								
0	1	0 0 0 0 0	R0	Horizontal Total *	Character	×	○								
0	1	0 0 0 0 1	R1	Horizontal Displayed	Character	×	○								
0	1	0 0 0 1 0	R2	Horizontal Sync* Position	Character	×	○								
0	1	0 0 0 1 1	R3	Horizontal Sync Width	Character	×	○								
0	1	0 0 1 0 0	R4	Vertical Total *	Line	×	○								
0	1	0 0 1 0 1	R5	Vertical Total Adjust	Raster	×	○								
0	1	0 0 1 1 0	R6	Vertical Displayed	Line	×	○								
0	1	0 0 1 1 1	R7	Vertical Sync * Position	Line	×	○								
0	1	0 1 0 0 0	R8	Interlace mode	–	×	○							V	S
0	1	0 1 0 0 1	R9	Maximum Raster Address	Raster	×	○								
0	1	0 1 0 1 0	R10	Cursor Start Raster	Raster	×	○		B	P					
0	1	0 1 0 1 1	R11	Cursor End Raster	Raster	×	○								
0	1	0 1 1 0 0	R12	Start Address(H)	–	×	○								
0	1	0 1 1 0 1	R13	Start Address(L)	–	×	○								
0	1	0 1 1 1 0	R14	Cursor(H)	–	○	○								
0	1	0 1 1 1 1	R15	Cursor (L)	–	○	○								
0	1	1 0 0 0 0	R16	Light Pen(H)	–	○	×								
0	1	1 0 0 0 1	R17	Light Pen(L)	–	○	×								

(NOTE) 1. The register marked *: (Written Value) = (Specified Value) – 1
2. R6 is programmed to be N/2 (N: total number of lines) in interlace sync & video mode.
3. When S is "1", V specifies video mode. S specifies the interlace sync mode.
4. B specifies the cursor blink. P specifies the cursor blink period.
5. The cursor start and the cursor end raster registers must be both even or both odd in interlace sync & video mode.
6. R0 is ordinally programmed to be odd number in interlace mode.
7. R9 must be programmed to be odd number in interlace sync & video mode.
8. ○; Yes, ×; No

■ FUNCTION OF INTERNAL REGISTERS

• Address Register (AR)

This is a 5-bit register used to select 18 internal control registers (R0~R17). Its contents are the address of one of 18 internal control registers. Programming the data from 18 to 31 produces no results. Access to R0~R17 requires, first of all, to write the address of corresponding control register into this register. When RS and $\overline{CS}$ are at "Low" level, this register is selected.

• Horizontal Total Register (R0)

This is a register used to program total number of horizontal characters per line including the retrace period. The data is 8-bit and its value should be programmed according to the specification of the CRT. When M is total number of characters, M-1 shall be programmed to this register. When programming for interlace mode, M must be even.

• Horizontal Displayed Register (R1)

This is a register used to program the number of horizontal displayed characters per line. Data is 8-bit and any number that is smaller than that of horizontal total characters can be programmed.

• Horizontal Sync Position Register (R2)

This is a register used to program horizontal sync position as multiples of the character clock period. Data is 8-bit and any number that is under the following condition (horizontal sync position + horizontal pulse width < horizontal total characters) can be programmed. When H is character number of horizontal sync position, H-1 shall be programmed to this register. When programmed value of this register is increased, the display position on the CRT screen is shifted to the left. When programmed value is decreased, the position is shifted to the right. Therefore, the optimum horizontal position can be determined by this value.

• Horizontal Sync Width Register (R3)

This is a register used to program the horizontal sync pulse width. The horizontal sync pulse width is programmed in the lower 4-bit as multiples of the character clock period. "0" can't be programmed.

• Vertical Total Register (R4)

This is a register used to program total number of lines per frame including vertical retrace period. The data is within 7-bit and its value should be programmed according to the specification of the CRTC. When N is total number of lines, N-1 shall be programmed to this register.

• Vertical Total Adjust Register (R5)

This is a register used to program the optimum number to adjust total number of rasters per field. This register enables to decide the number of vertical deflection frequency more strictly.

• Vertical Displayed Register (R6)

This is a register used to program the number of displayed character rows on the CRT screen. Data is 7-bit and any number that is smaller than that of vertical total characters can be programmed.

Table 2 Pulse Width of Horizontal Sync Signal

HSW				Pulse Width
2^3	2^2	2^1	2^0	
0	0	0	0	– (Note)
0	0	0	1	1 CH
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	10
1	0	1	1	11
1	1	0	0	12
1	1	0	1	13
1	1	1	0	14
1	1	1	1	15

CH; Character clock period
(Note) HSW = "0" can't be used.

• Vertical Sync Position Register (R7)

This is a register used to program the vertical sync position on the screen as multiples of the horizontal character line period. Data is 7-bit and any number that is equal to or less than vertical total characters can be programmed. When V is character number of vertical sync position, V-1 shall be programmed to this register. When programmed value of this register is increased, the display position is shifted up. When programmed value is decreased, the position is shifted down. Therefore, the optimum vertical position may be determined by this value.

• Interlace mode Register (R8)

This is a register used to program raster scan mode.

Raster Scan Mode Program Bit (V, S)

Raster scan mode is programmed in the V, S bit.

Table 3 Raster Scan Mode (2^1, 2^0)

V	S	Raster Scan Mode
0	0	Non-interlace Mode
1	0	
0	1	Interlace Sync Mode
1	1	Interlace Sync & Video Mode

In the non-interlace mode, the rasters of even number field and old number field are scanned duplicatedly. In the interlace sync mode, the rasters of odd number field are scanned in the middle of even number field. Then it is controlled to display the same character pattern in two fields. In the interlace sync & video mode, the raster scan method is the same as the interlace sync mode, but it is controlled to display different character pattern in two field.

- **Maximum Raster Address Register (R9)**

This is a register used to program maximum raster address within 5-bit. This register defines total number of rasters per character including line space. When total number of rasters is RN, RN-1 shall be programmed to this register. Moreover, RN must be even in interlace sync & video mode.

- **Cursor Start Raster Register (R10)**

This is a register used to program the cursor start raster address by lower 5-bit (2^0~2^4) and the cursor display mode by higher 2-bit (2^5, 2^6).

Table 4 Cursor Display Mode (2^6, 2^5)

B	P	Cursor Display Mode
0	0	Non-blink
0	1	Cursor Non-display
1	0	Blink 16 Field Period
1	1	Blink 32 Field Period

- **Cursor End Raster Register (R11)**

This is register used to program the cursor end raster address, both the cursor start raster register and the cursor end raster register must be even or odd in interlace sync & video mode.

- **Start Address Register (R12, R13)**

These are used to program the first address of refresh memory to read out.

Paging and scrolling is easily performed using this register.

- **Cursor Register (R14, R15)**

These two read/write registers stores the cursor location. The higher 2-bit (2^6, 2^7) of R14 are always "0".

- **Light Pen Register (R16, R17)**

These read only registers are used to catch the detection address of the light pen. The higher 2-bit (2^6, 2^7) of R16 are always "0". Its value needs to be corrected by software because there is time delay from address output of the CRTC to signal input LPSTB pin of the CRTC in the process that raster is lit after address output and light pen detects it. Moreover, delay time shown in Fig. 2 needs to be taken into account.

Restriction on Programming Internal Register

i) $0 < Nhd < Nht + 1 \leq 256$
ii) Nht : Odd Number (in the case of interlace sync mode or interlace sync & video mode)
iii) $0 < Nvd < Nvt + 1 \leq 128$
iv) $0 < Nhsp + Nhsw < Nht + 1$
v) $0 \leq Nvsp \leq Nvt$
vi) $0 \leq N_{CSTART} \leq N_{CEND} \leq Nr$
vii) In the case of interlace sync & video mode, either of the followings shall be selected.
  (1) N_{CSTART}, N_{CEND} : Even Number (The cursor is displayed on the even raster.)
  (2) N_{CSTART}, N_{CEND} : Odd Number (The cursor is displayed on the odd raster.)
viii) $0 \leq Nhd \leq Nhsp$

Notes for Use

The method of directly using the value programmed in the internal registers of LSI for controlling the CRT is adopted. Consequently, the display may flicker on the screen when the contents of the registers are changed from bus side asynchronously with the display operation.

Cursor Register

Writing into this register at frequent intervals for moving the cursor should be performed during horizontal and vertical retrace period.

Start Address Register

Writing into the start address register at frequent intervals for scrolling and paging should be performed during horizontal and vertical display period.

It is desirable to avoid programming other registers during display operation.

■ OPERATION OF THE CRTC

- **Time Chart of CRT Interface Signals**

The following example shows the display operation in which values of Table 5 are programmed to the CRTC internal registers. Fig. 7 shows the CRT screen format. Fig. 10 shows the time chart of signals output from the CRTC.

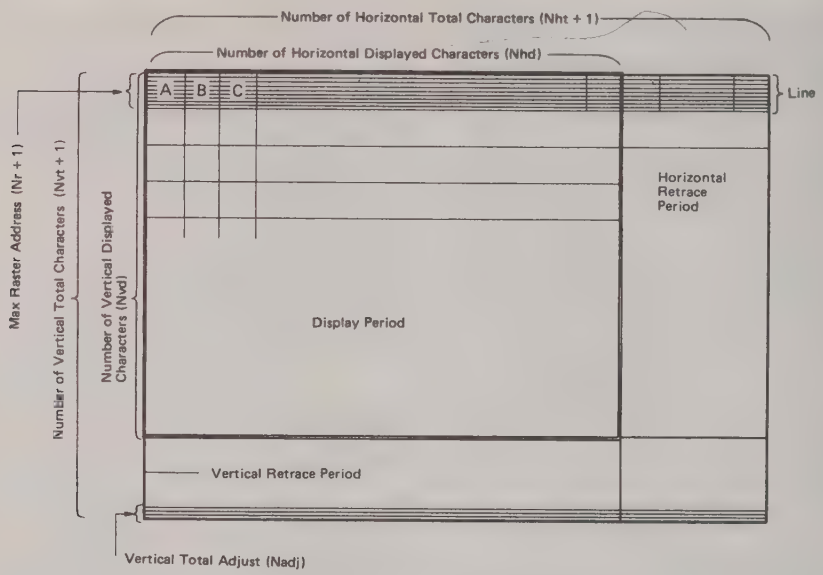


Figure 7 CRT Screen Format

Table 5 Programmed Values into the Registers

Register	Register Name	Value	Register	Register Name	Value
R0	Horizontal Total	Nht	R9	Max Raster Address	Nr
R1	Horizontal Displayed	Nhd	R10	Cursor Start Raster	
R2	Horizontal Sync Position	Nhsp	R11	Cursor End Raster	
R3	Horizontal Sync Width	Nhsw	R12	Start Address (H)	0
R4	Vertical Total	Nvt	R13	Start Address (L)	0
R5	Vertical Total Adjust	Nadj	R14	Cursor (H)	
R6	Vertical Displayed	Nvd	R15	Cursor (L)	
R7	Vertical Sync Position	Nvsp	R16	Light Pen (H)	
R8	Interlace mode		R17	Light Pen (L)	

[NOTE] Nhd<Nht, Nvd<Nvt

The relation between values of Refresh Memory Address ($MA_0 \sim MA_{13}$) and Raster Address ($RA_0 \sim RA_4$) and the display position on the screen is shown in Fig. 16. Fig. 16 shows the case where the value of Start Address is 0.

- **Interlace Control**

Fig. 8 shows an example where the same character is displayed in the non-interlace mode, interlace sync mode, and interlace sync & video mode.

Non-interlace Mode Control

In non-interlace mode, each field is scanned duplicatedly. The values of raster addresses ($RA_0 \sim RA_4$) are counted up one from 0.

Interlace Sync Mode Control

In the interlace sync mode, raster addressed in the even field and the odd field are the same as addressed in the non-interlace mode. One character pattern is displayed mutually and its displayed position in the odd field is set at 1/2 raster space down from that in the even field.

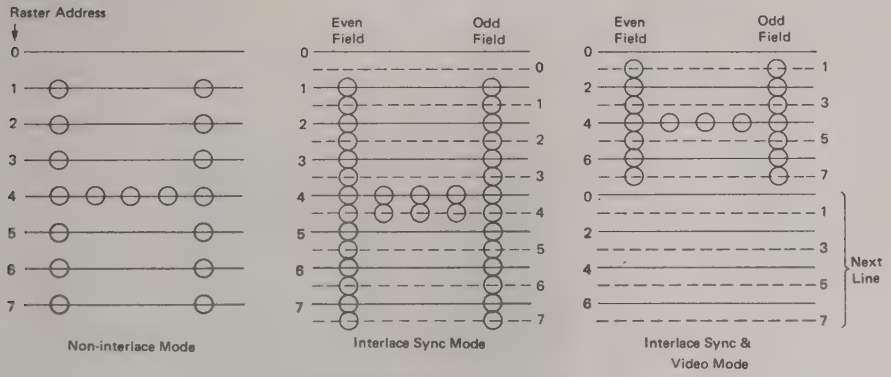


Figure 8 Example of Raster Scan Display

Interlace Sync & Video Mode Control

In the interlace sync & video mode, the character pattern of even number raster address is displayed in even field. Also odd number raster address is displayed in odd field.

- **Cursor Control**

Fig. 9 shows the display patterns where each value is programmed to the cursor start raster register and the cursor end raster register. Programmed values to the cursor start raster register and the cursor end raster register need to be under the following condition.

Cursor Start Raster Register ≦ Cursor End Raster Register ≦ Maximum Raster Address Register.

Time chart of CUDISP is shown in Fig. 14 and Fig. 15.

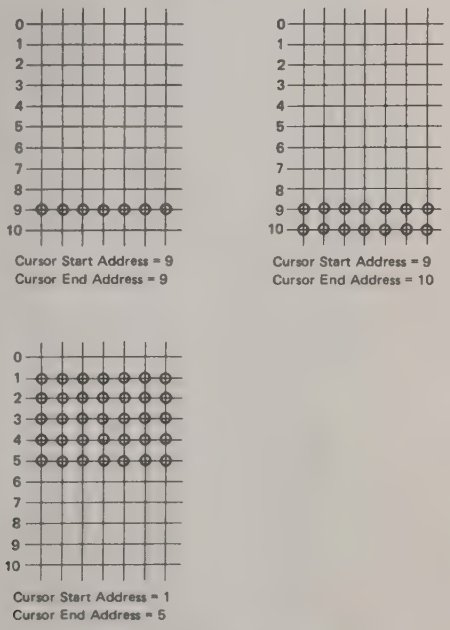

Cursor Start Address = 9
Cursor End Address = 9
Cursor Start Address = 9
Cursor End Address = 10
Cursor Start Address = 1
Cursor End Address = 5

Figure 9 Cursor Control

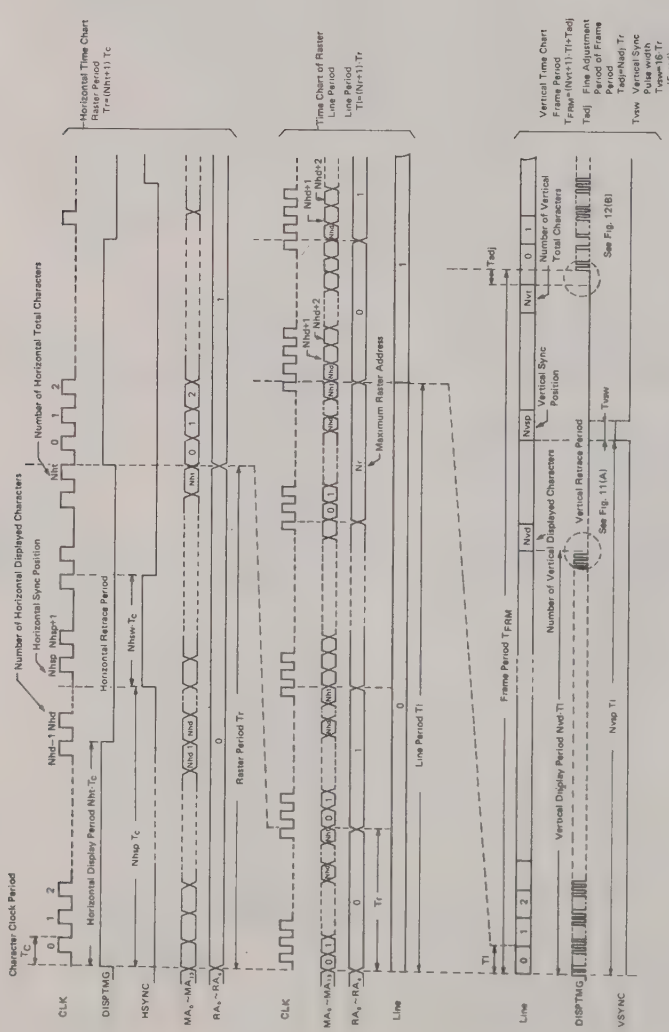


Figure 10 CRTC Time Chart

Output waveform of horizontal & vertical display in the case where values shown in Table 5 are programmed to each register.

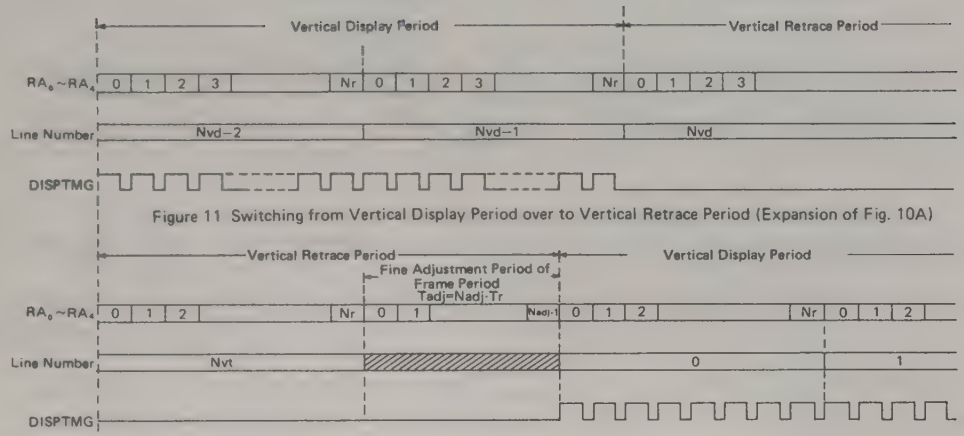
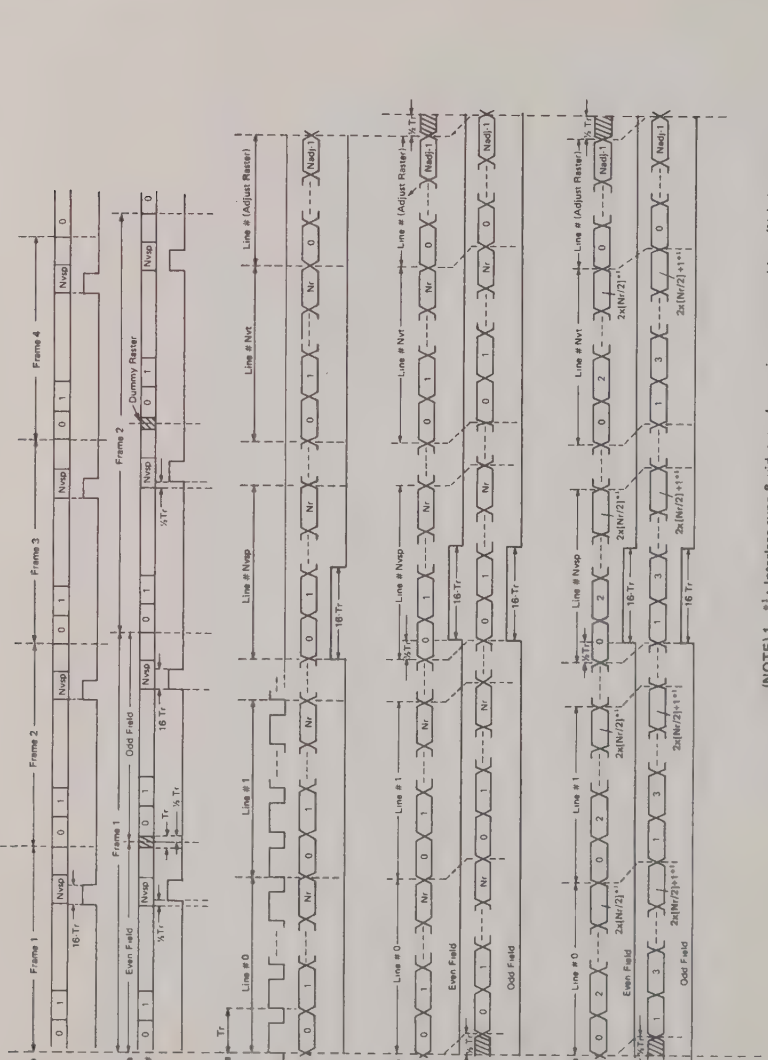


Figure 11 Switching from Vertical Display Period over to Vertical Retrace Period (Expansion of Fig. 10A)

Figure 12 Fine Adjustment Period of Frame in Vertical Display
(Expansion of Fig. 10B)



(NOTE) 1. *1: Interlace sync & video mode, maximum raster address (Nr) shall be odd.
2. In interlace mode, Nht shall be odd.

Figure 13 Interlace Control

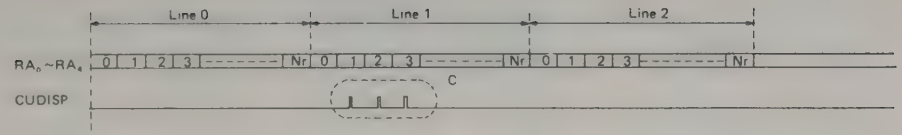
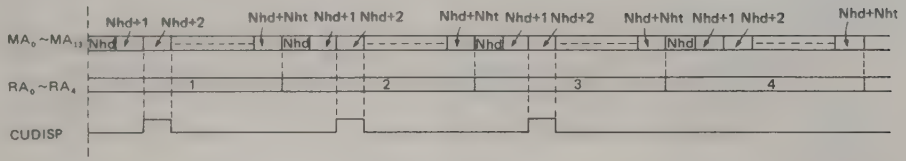


Figure 14 Relation between Line · Raster and CUDISP



Cursor register = Nhd+2
Cursor Start
Raster Register = 1
Cursor End
Raster Register = 3
are Programmed in cursor display mode.

In blink mode, it is changed into display or non-display mode when field period is 16 or 32-time period.

Figure 15 CUDISP Timing (Expansion of Fig. 14C)

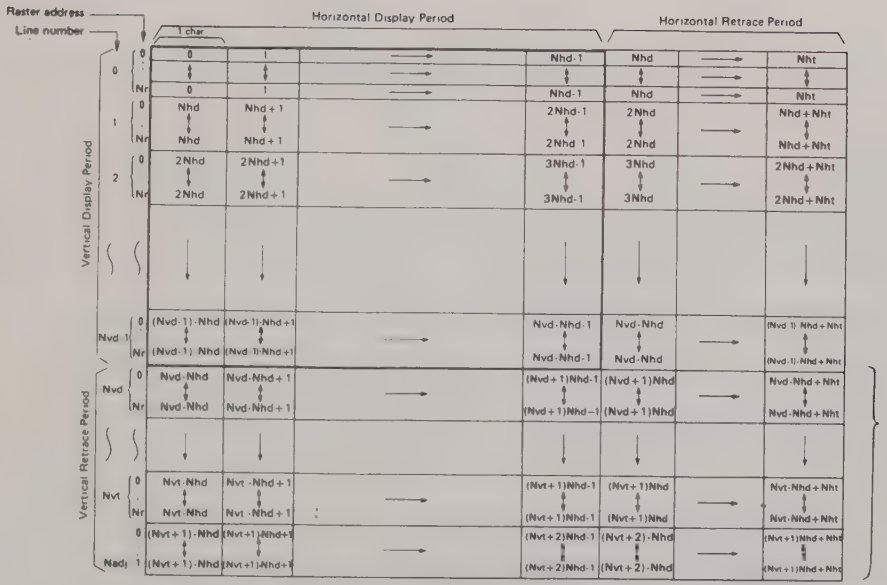

Raster address
Line number
1 char
Horizontal Display Period
Horizontal Retrace Period
Vertical Display Period
Vertical Retrace Period
Valid refresh memory address (0~Nvd·Nhd-1) are shown within the thick-line square.
Refresh memory address are provided even during horizontal and Vertical retrace period.
This is an example in the case where the programmed value of start address register is 0.

Figure 16 Refresh Memory Address (MA_0~MA_{13})

■ How to Use the CRTC

● Interface to MPU

As shown in Fig. 17, the CRTC is connected with the standard bus of MPU to control the data transfer between them. The CRTC address is determined by $\overline{CS}$ and RS, and the Read/Write operation is controlled by R/$\overline{W}$ and E. When $\overline{CS}$ is "Low" and RS is also "Low", the CRTC address register is selected. When $\overline{CS}$ is "Low" and RS is "High", one of 18 internal registers is selected.

$\overline{RES}$ is the system reset signal. When $\overline{RES}$ becomes "Low", the CRTC internal control logic is reset. But internal registers shown in Table 1 (R0~R17) are not affected by $\overline{RES}$ and remain unchanged.

The CRTC is designed so as to provide an interface to microcomputers, but adding some external circuits enables an interface to other data sources.

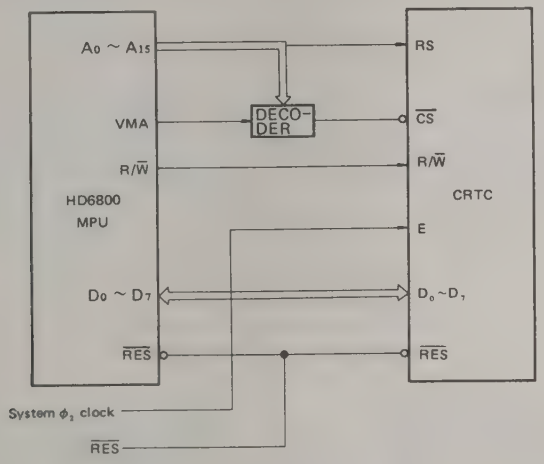


Figure 17 Interface to MPU

● Dot Timing Generating Circuit

CRTC's CLK input (21 pin) is provided with CLK which defines horizontal character time period from the outside. This CLK is generated by dot counter shown in Fig. 18. Fig. 18 shows a example of circuit where horizontal dot number of the character is "9". Fig. 19 shows the operation time chart of dot counter shown in Fig. 18. As this example shows explicitly, CLK is at "Low" level in the former half of horizontal character time and at "High" level in the latter half. It is necessary to be careful so as not to mistake this polarity.

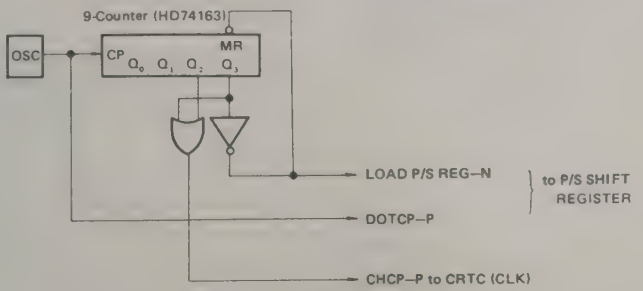


Figure 18 Example of Dot Counter

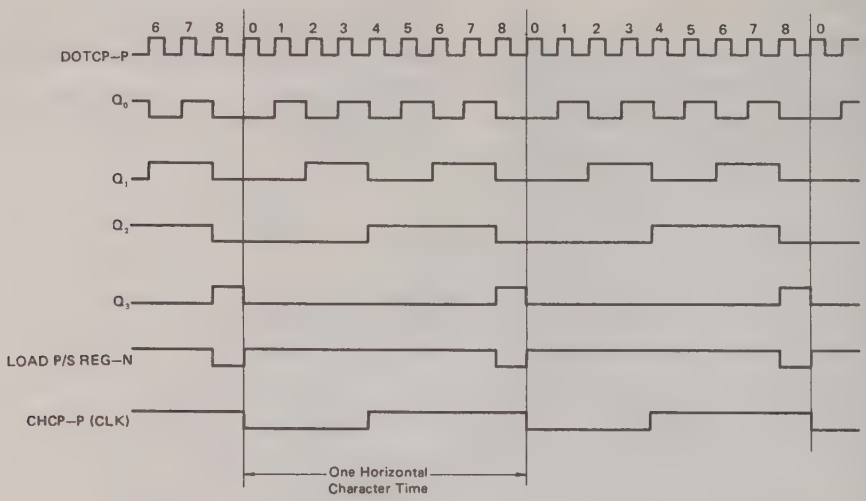


Figure 19 Time Chart of Dot Counter

■ INTERFACE TO DISPLAY CONTROL UNIT

Fig. 20 shows the interface between the CRTC and display control unit. Display control unit is mainly composed of Refresh Memory, Character Generator, and Video Control circuit. For refresh memory, 14 Memory Address line (0~16383) max are provided and for character generator, 5 Raster Address line (0~31) max are provided. For video control circuit, DISPTMG, CUDISP, HSYNC, and VSYNC are sent out. DISPTMG is used to control the blank period of video signal. CUDISP is used as video signal to display the cursor on the CRT screen. Moreover, HSYNC and VSYNC are used as drive signals respectively for CRT horizontal and vertical deflection circuits.

Outputs from video control circuit, (video signals and sync signals) are provided to CRT display unit to control the deflection and brightness of CRT, thus characters are displayed on the screen.

Fig. 21 shows more detailed block diagram of display control unit. This shows how to use DISPTMG and CUDISP signals. By delaying for one or two-character time. DISPTMG signal synchronizes with output timing of parallel-serial converter to control blanking of character video signal. By delaying for one or two-character time. CUDISP signal is mixed with character video signal. Whether delay time of DISPTMG and CUDISP signals should be one or two-character time, moreover, whether LATCH register is necessary for output from refresh memory or not are determined as shown in Fig. 21 by the relations among one horizontal character time, delay time of Memory Address, access time of refresh memory, and access time of character generator.

For reference, time chart from refresh memory address MA to video signal in the case of two-character time delay is shown in Fig. 22.

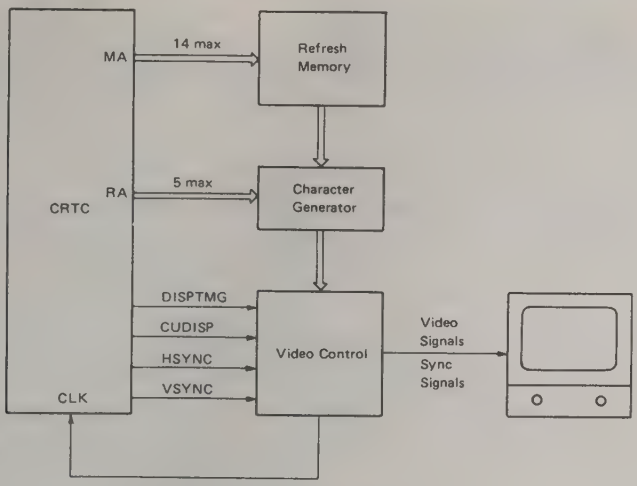
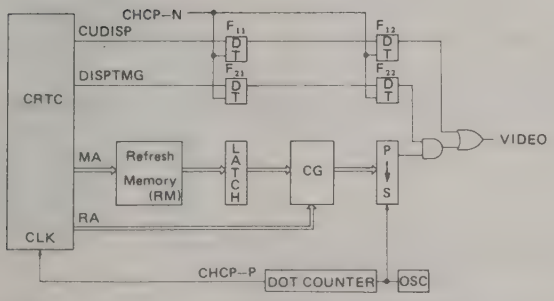


Figure 20 Interface to Display Control Unit



Case	Access Time of RM and CG	LATCH	F_{11}	F_{21}	F_{12}	F_{22}
1	RM Access + CG Access > $t_{CH} - t_{MAD}$	N	N	N	N	N
2	RM Access + CG Access < $t_{CH} - t_{MAD}$	UN	N	N	UN	UN

[NOTE] N : Necessary
UN : Unnecessary
t_{CH} : Cycle time of CHCP
t_{MAD} : Delay time of MA
RM : Refresh Memory

Figure 21 Display Control Unit

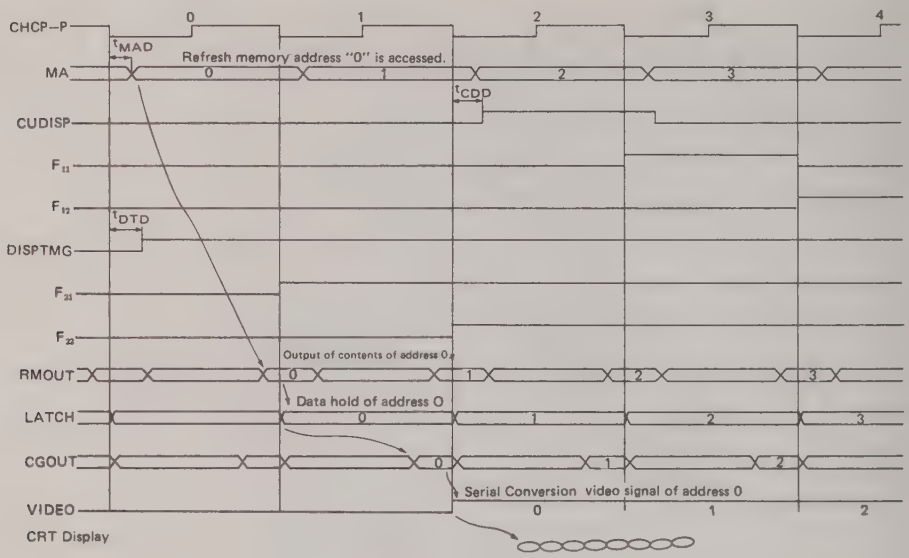


Figure 22 Time Chart of Display Control Unit

■ HOW TO DECIDE PARAMETERS SET ON THE CRTC

• How to Decide Parameters Based on Specification of CRT Display Unit (Monitor)

Number of Horizontal Total Characters

Horizontal deflection frequency f_h is given by specification of CRT display unit. Number of horizontal total characters is determined by the following equation.

$$f_h = \frac{1}{t_C\,(Nht + 1)}$$

where,

t_C : Cycle Time of CLK (Character Clock)
Nht : Programmed Value of Horizontal Total Register (R0)

Number of Vertical Total Characters

Vertical deflection frequency is given by specification of CRT display unit. Number of vertical Total characters is determined by the following equation.

1) Non-interlace Mode
$Rt = (Nvt + 1)(Nr + 1) + Nadj$
2) Interlace Mode
$Rt = (Nvt + 1)(Nr + 1) + Nadj + 0.5$

where,

Rt : Number of Total Rasters per frame (Including retrace period)
Nvt : Programmed Value of Vertical Total Register (R4)
Nr : Programmed Value of Maximum Raster Address Register (R9)
Nadj : Programmed Value of Vertical Total Adjust Register (R5)

Horizontal Sync Pulse Width

Horizontal sync pulse width is programmed to low order 4-bit of horizontal sync width register (R3) in unit of horizontal character time. Programmed value can be selected within from 1 to 15.

Horizontal Sync Position

As shown in Fig. 23, horizontal sync position is normally selected to be in the middle of horizontal retrace period. But there are some cases where is optimum sync position is not located in the middle of horizontal retrace period according to specification of CRT. Therefore, horizontal sync position should be determined by specification of CRT. Horizontal sync pulse position is programmed in unit of horizontal character time.

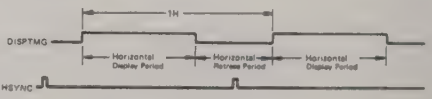


Figure 23 Time Chart of HSYNC

Vertical Sync Position

As shown in Fig. 24, vertical sync position is normally selected to be in the middle of vertical retrace period. But there are some cases where its optimum sync position is not located in the middle of vertical retrace period according to specification of CRT. Therefore, vertical sync position should be determined by specification of CRT. Vertical sync pulse position is programmed to vertical sync position register (R7) in unit of line period.

- **How to Decide Parameters Based on Screen Format**

Dot Number of Characters (Horizontal)

Dot number of characters (horizontal) is determined by character font and character space. An example is shown in Fig. 25. More strictly, dot number of characters (horizontal) N is determined by external N-counter. Character space is set by means shown in Fig. 26.

Dot Number of Characters (Vertical)

Dot number of characters (vertical) is determined by characters font and line space. An example is shown in Fig. 25. Dot number of characters (vertical) is programmed to maximum raster address (R9) of CRTC.

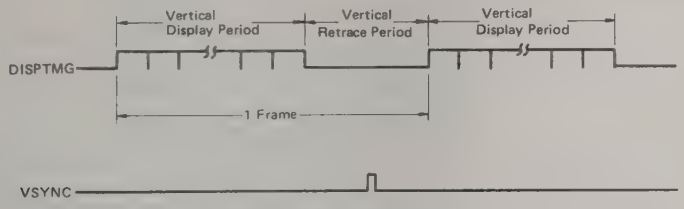


Figure 24 Time Chart of VSYNC

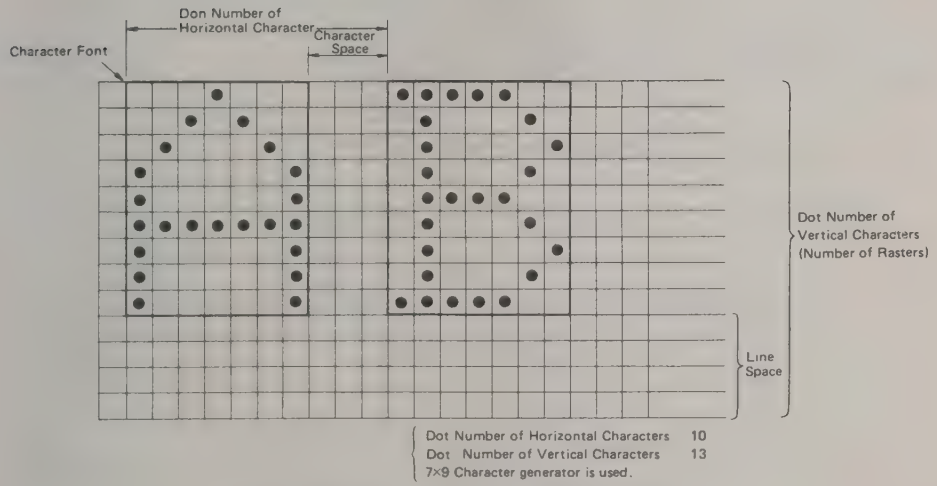


Figure 25 Dot Number of Horizontal and Vertical Characters

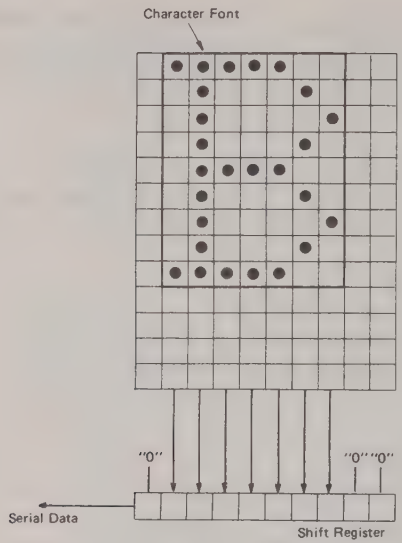


Figure 26 How to Make Character Space

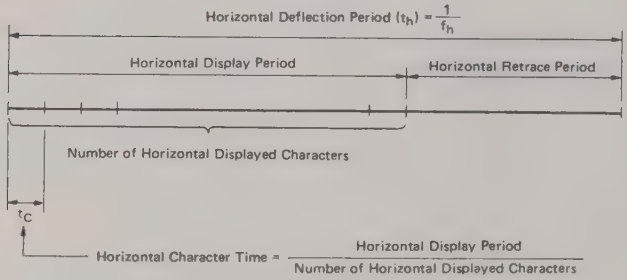

Horizontal Deflection Period (t_h) = $\frac{1}{f_h}$
Horizontal Display Period
Horizontal Retrace Period
Number of Horizontal Displayed Characters
t_C
Horizontal Character Time = $\frac{\text{Horizontal Display Period}}{\text{Number of Horizontal Displayed Characters}}$

Figure 27 Number of Horizontal Displayed Characters

Number of Horizontal Displayed Characters

Number of horizontal displayed characters is programmed to horizontal displayed register (R1) of the CRTC. Programmed value is based on screen format. Horizontal display period, which is given by specification of horizontal deflection frequency and horizontal retrace period of CRT display unit, determines horizontal character time, being divided by number of horizontal displayed characters. Moreover, its cycle time and access time which are necessary for CRT display system are determined by horizontal character time.

Number of Vertical Displayed Characters

Number of vertical displayed characters is programmed to vertical displayed register (R6). Programmed value is based on screen format. As specification of vertical deflection frequency of CRT determines number of total rasters (Rt) including vertical retrace period and the relation between number of vertical displayed character and total number of rasters on a screen is as mentioned above, CRT which is suitable for desired screen format should be selected.

For optimum screen format, it is necessary to adjust number of rasters per line, number of vertical displayed characters, and total adjust raster (Nadj) within specification of vertical deflection frequency.

Scan Mode

The CRTC can program three-scan modes shown in Table 6 to interlace mode register (R8). An example of character display in each scan mode is shown in Fig. 8.

Table 6 Program of Scan Mode

V	S	Scan Mode	Main Usage
0	0	Non-interlace	Normal Display of Characters & Figures
1	0		
0	1	Interlace Sync	Fine Display of Characters & Figures
1	1	Interlace Sync & Video	Display of Many Characters & Figures Without Using High-resolution CRT

[NOTE] In the interlace mode, the number of times per sec. in raster scanning on one spot on the screen is half as many as that in non-interlace mode. Therefore, when persistence of luminescence is short, flickering may happen. It is necessary to select optimum scan mode for the system, taking characteristics of CRT, raster scan speed, and number of displayed characters and figures into account.

Cursor Display Method

Cursor start raster register and cursor end raster register (R10, R11) enable programming the display modes shown in Table 4 and display patterns shown in Fig. 9. Therefore, it is possible to change the method of cursor display dynamically according to the system conditions as well as to realize the cursor display that meets the system requirements.

Start Address

Start address resisters (R12, R13) give an offset to the address of refresh memory to read out. This enables paging and scrolling easily.

Cursor Register

Cursor registers (R14, R15) enable programming the cursor display position on the screen. As for cursor address, it is not X, Y address but linear address that is programmed.

■ Applications of the CRTC

● Monochrome Character Display

Fig. 28 shows a system of monochrome character display. Character clock signal (CLK) is provided to the CRTC through OSC and dot counter. It is used as basic clock which drives internal control circuits. MPU is connected with the CRTC by standard bus and controls the CRTC initialization and read/write of internal registers.

Refresh memory is composed of RAM which has capacity of one frame at least and the data to be displayed is coded and stored. The data to refresh memory is changed through MPU

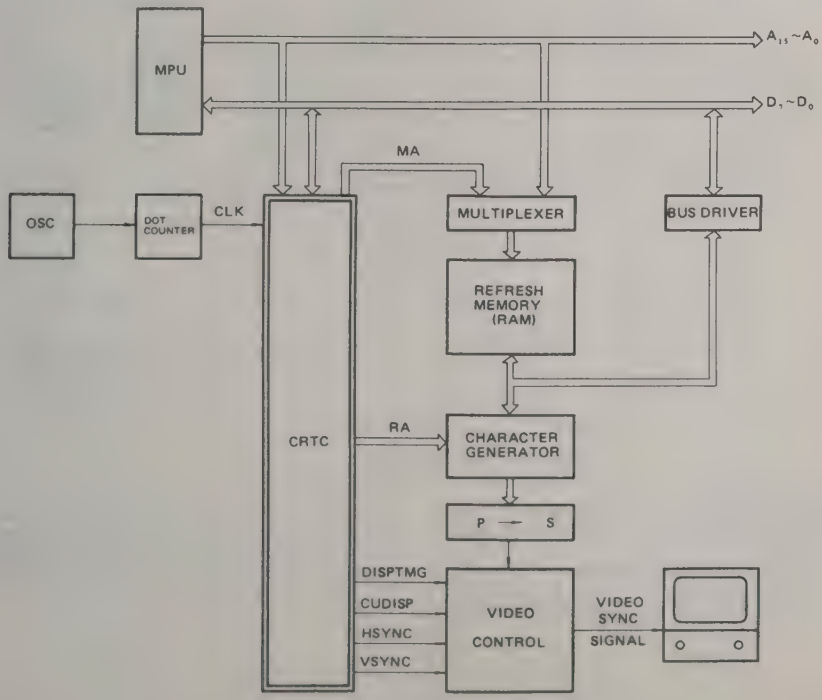


Figure 28 Monochrome Character Display

bus, while refresh memory is read out successively by the CRTC to display a static pattern on the screen. Refresh memory is accessed by both MPU and the CRTC, so it needs to change its address selectively by multiplexer. The CRTC has 14 MA (Memory Address output), but in fact some of them that are needed are used according to capacity of refresh memory.

Code output of refresh memory is provided to character generator. Character generator generates a dot pattern of a specified raster of a specified character in parallel according to code output from refresh memory and RA (Raster Address output) from the CRTC. Parallel-serial converter is normally composed of shift register to convert output of character generator into a serial dot pattern. Moreover, DISPTMG, CUDISP, HSYNC, and VSYNC are provided to video control circuit. It controls blanking for output of parallel-serial converter, mixes these signals with cursor video signal, and generates sync signals for an interface to monitor.

- **Color Character Display**

Fig. 29 shows a system of color character display. In this example, a 3-bit color control bit (R, G, B) is added to refresh memory in parallel with character code and provided to video control circuit. Video control circuit controls coloring as well as blanking and provides three primary color video signals (R, G, B signals) to CRT display device to display characters in seven kinds of color on the screen.

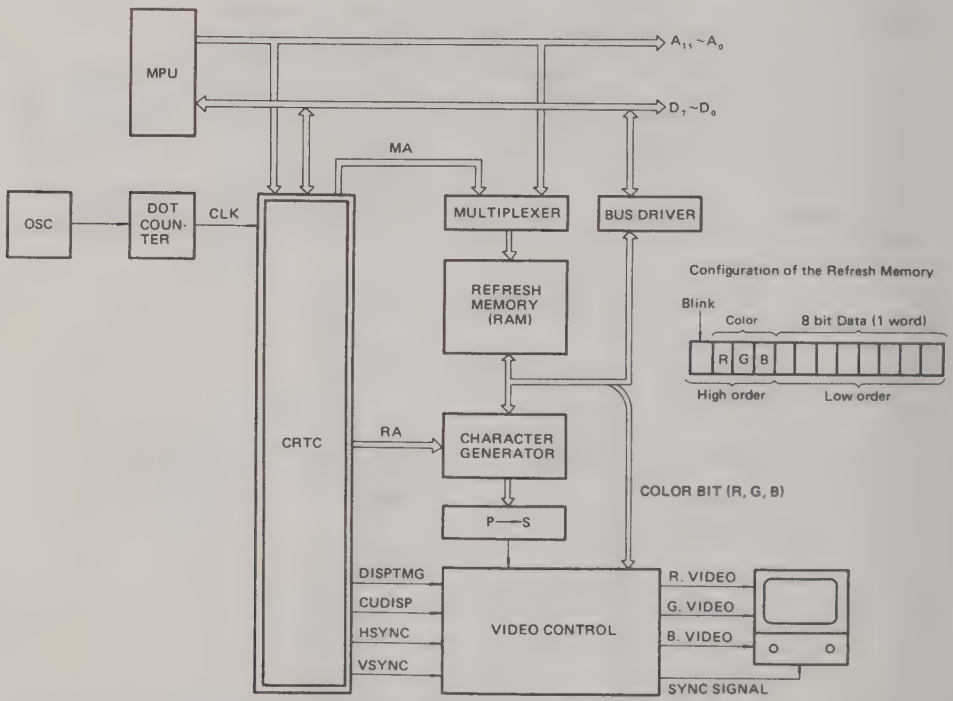


Figure 29 Color Character Display

- **Color limited Graphic Display**

Limited graphic display is to display simple figures as well as character display by combination of picture element which are defined in unit of one character.

As shown in Fig. 30, graphic pattern generator is set up in parallel with character generator and output of these generators are wire-ORed. Which generator is accessed depends on coded output of refresh memory.

In this example, graphic pattern generator adopts ROM, so only the combination of picture elements which are programmed to it is used for this graphic display system. Adopting RAM instead of ROM enables dynamically writable symbols in any combination on one display by changing the contents of them.

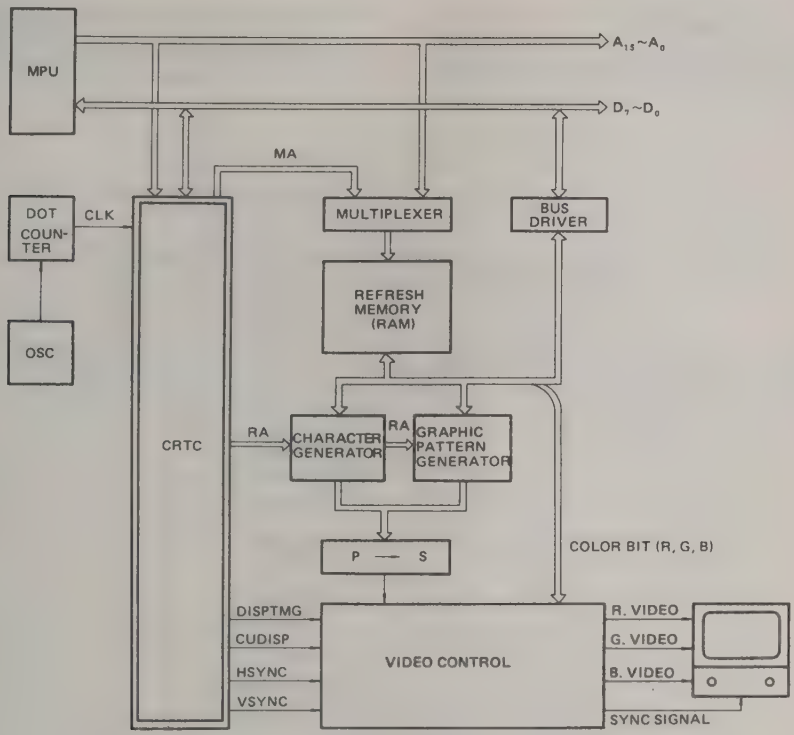


Figure 30 Color Limited Graphic Display

- **Monochrome Full Graphic Display**

Fig. 31 shows a system of monochrome full graphic display. While simple graphic display is figure display by combination of picture elements in unit of 1 picture elements, full graphic display is display of any figures in unit of 1 dot. In this case, refresh memory is dot memory that stores all the dot patterns, so its output is directly provided to parallel-serial converter to be displayed. Dot memory address to refresh the screen is set up by combination of MA and RA of CRTC.

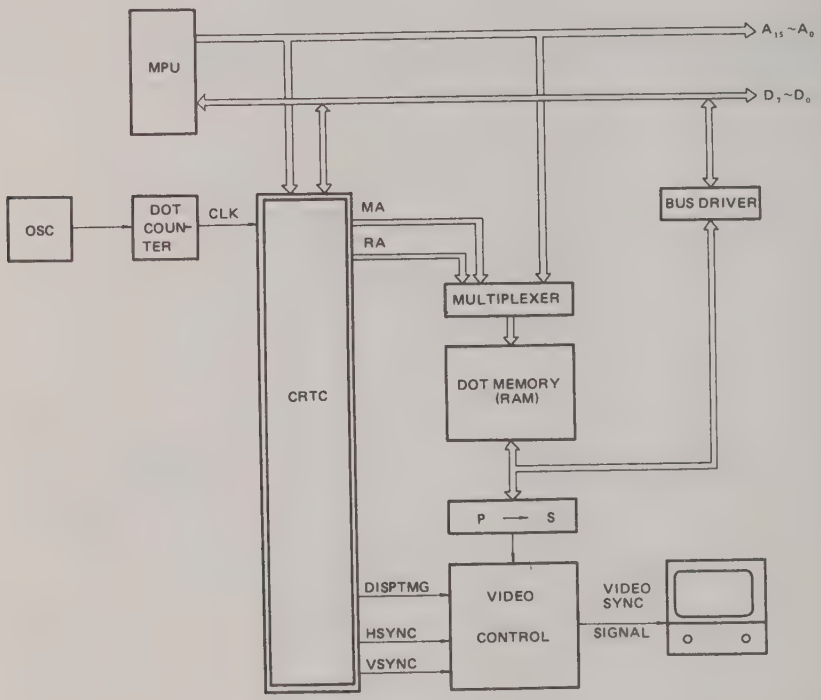


Figure 31 Monochrome Full Graphic Display

Fig. 32 shows an example of access to refresh memory by combination of MA and RA. Fig. 32 shows a refresh memory address method for full graphic display. Correspondence between dot on the CRT screen and refresh memory address shown in Fig. 33.

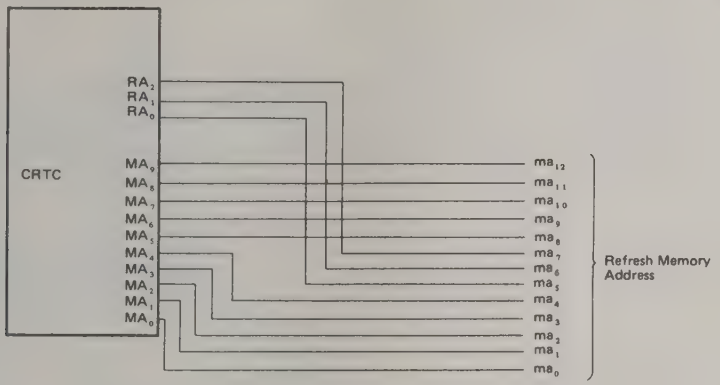


Figure 32 Refresh Memory Address Method for Full Graphic Display

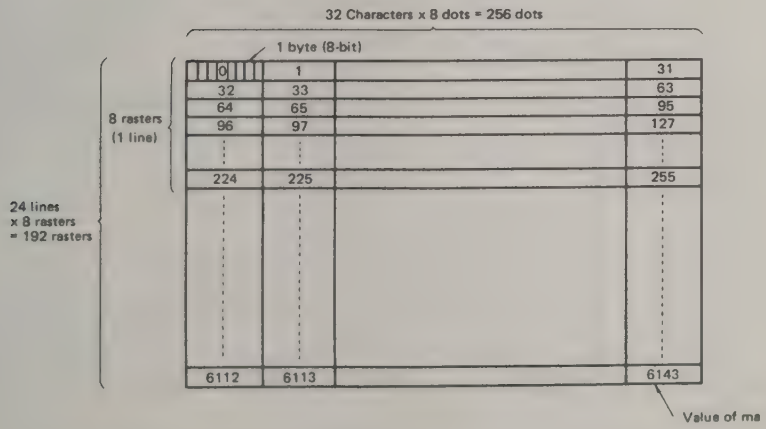


Figure 33 Memory Address and Dot Display Position on the Screen for Full Graphic Display

- **Color Full Graphic Display**

Fig. 34 shows a system of color full graphic display by 7-color display. Refresh memory is composed of three dot memories which are respectively used for red, green, and blue. These dot memories are read out in parallel at one time and their output is provided to three parallel-serial converters. Then video control circuit adds the blanking control to output of these converters and provides it to CRT display device as red, green, and blue video signals with sync signals.

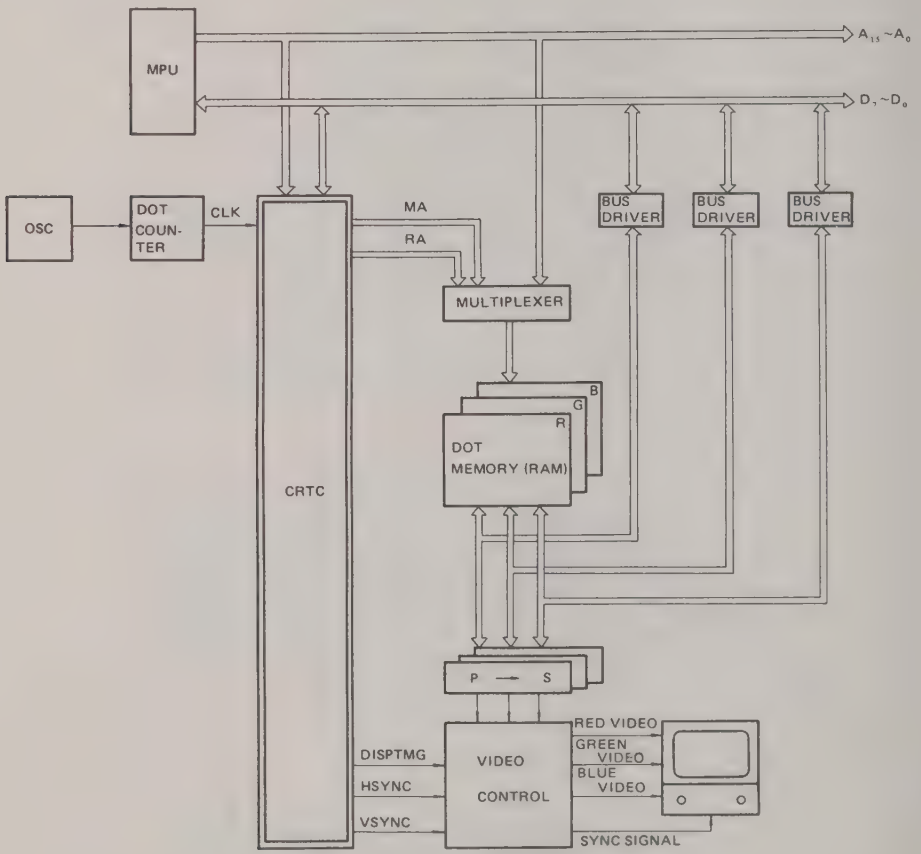


Figure 34 Color Full Graphic Display

- **Cluster Control of CRT Display**

The CRTC enables cluster control that is to control CRT display of plural devices by one CRTC. Fig. 35 shows a system of cluster control. Each display control unit has refresh memory, character generator, parallel-serial converter, and video control circuit separately, but these are controlled together by the CRTC.

In this system, it is possible for plural CRT display devices to have their own display separately.

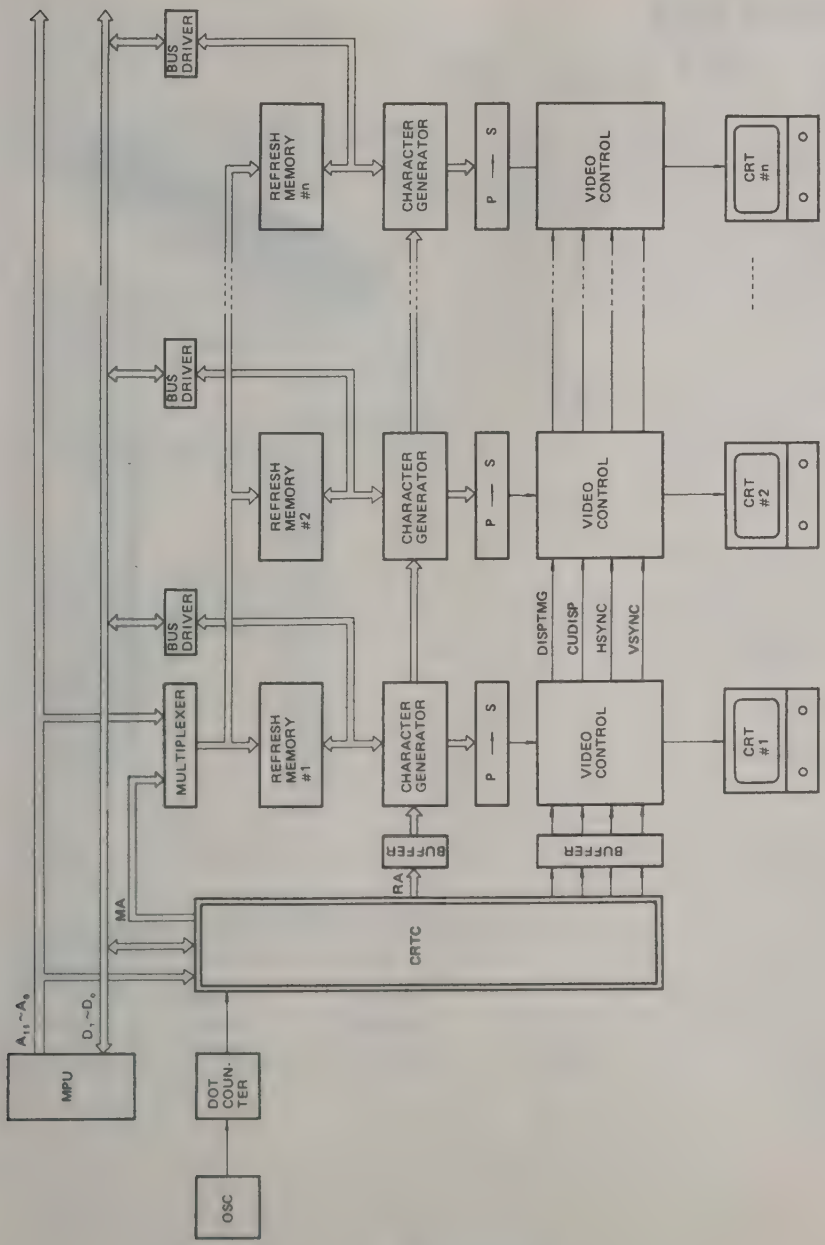


Figure 35 Cluster Control by the CRTC

HD6845S, HD68A45S, HD68B45S

CRTC (CRT Controller)

The CRTC is a LSI controller which is designed to provide an interface for microcomputers to raster scan type CRT displays. The CRTC belongs to the HMCS6800 LSI Family and has full compatibility with MPU in both data lines and control lines. Its primary function is to generate timing signal which is necessary for raster scan type CRT display according to the specification programmed by MPU. The CRTC is also designed as a programmable controller, so applicable to wide-range CRT display from small low-functioning character display up to raster type full graphic display as well as large high-functioning limited graphic display.

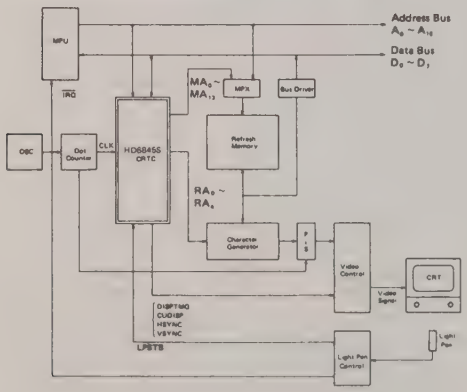
HD6845SP, HD68A45SP, HD68B45SP

(DP-40)

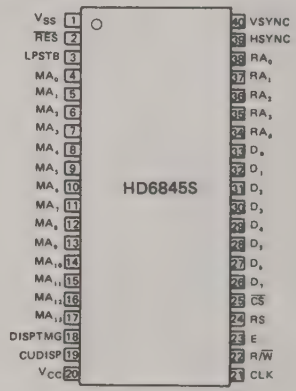
■ FEATURES

- Number of Displayed Characters on the Screen, Vertical Dot Format of One Character, Horizontal and Vertical Sync Signal, Display Timing Signal are Programmable
- 3.7 MHz High Speed Display Operation
- Line Buffer-less Refreshing
- 14-bit Refresh Memory Address Output (16k Words max. Access)
- Programmable Interlace/Non-interlace Scan Mode
- Built-in Cursor Control Function
- Programmable Cursor Height and its Blink
- Built-in Light Pen Detection Function
- Paging and Scrolling Capability
- TTL Compatible
- Single +5V Power Supply

■ SYSTEM BLOCK DIAGRAM



■ PIN ARRANGEMENT



(Top View)

■ ORDERING INFORMATION

CRTC	Bus Timing	CRT Display Timing
HD6845S	1.0 MHz	3.7 MHz max.
HD68A45S	1.5 MHz	
HD68B45S	2.0 MHz	

■ ABSOLUTE MAXIMUM RATINGS

Item	Symbol	Value	Unit
Supply Voltage	V_{CC} *	−0.3 ~ +7.0	V
Input Voltage	V_{in} *	−0.3 ~ +7.0	V
Operating Temperature	T_{opr}	− 20 ~ + 75	°C
Storage Temperature	T_{stg}	− 55 ~ +150	°C

* With respect to V_{SS} (SYSTEM GND)

[NOTE] Permanent LSI damage may occur if maximum ratings are exceeded. Normal operation should be under recommended operating conditions. If these conditions are exceeded, it could affect reliability of LSI.

■ RECOMMENDED OPERATING CONDITIONS

Item	Symbol	min	typ	max	Unit
Supply Voltage	V_{CC} *	4.75	5.0	5.25	V
Input Voltage	V_{IL} *	−0.3	–	0.8	V
	V_{IH} *	2.0	–	V_{CC}	V
Operating Temperature	T_{opr}	− 20	25	75	°C

* With respect to V_{SS} (SYSTEM GND)

■ ELECTRICAL CHARACTERISTICS

• DC CHARACTERISTICS (V_{CC} = 5V ± 5%, V_{SS} = 0V, Ta = −20~+75°C, unless otherwise noted.)

Item	Symbol	Test Condition		min	typ*	max	Unit
Input "High" Voltage	V_{IH}			2.0	–	V_{CC}	V
Input "Low" Voltage	V_{IL}			−0.3	–	0.8	V
Input Leakage Current	I_{in}	V_{in} = 0 ~ 5.25V (Except D_0~D_7)		−2.5	–	2.5	μA
Three-State Input Current (off-state)	I_{TSI}	V_{in} = 0.4 ~ 2.4V V_{CC} = 5.25V (D_0~ D_7)		− 10	–	10	μA
Output "High" Voltage	V_{OH}	I_{LOAD} = −205 μA (D_0~D_7)		2.4	–	–	V
		I_{LOAD} = −100 μA (Other Outputs)					
Output "Low" Voltage	V_{OL}	I_{LOAD} = 1.6 mA		–	–	0.4	V
Input Capacitance	C_{in}	V_{in} = 0 Ta = 25°C f = 1.0 MHz	D_0 ~ D_7	–	–	12.5	pF
			Other Inputs	–	–	10.0	pF
Output Capacitance	C_{out}	V_{in} = 0V, Ta = 25°C, f = 1.0 MHz		–	–	10.0	pF
Power Dissipation	P_D			–	600	1000	mW

* Ta = 25°C, V_{CC} = 5.0V

- **AC CHARACTERISTICS (V_{CC} = 5V ±5%, V_{SS} = 0V, Ta = -20~+75°C, unless otherwise noted.)**

1. TIMING OF CRTC SIGNAL

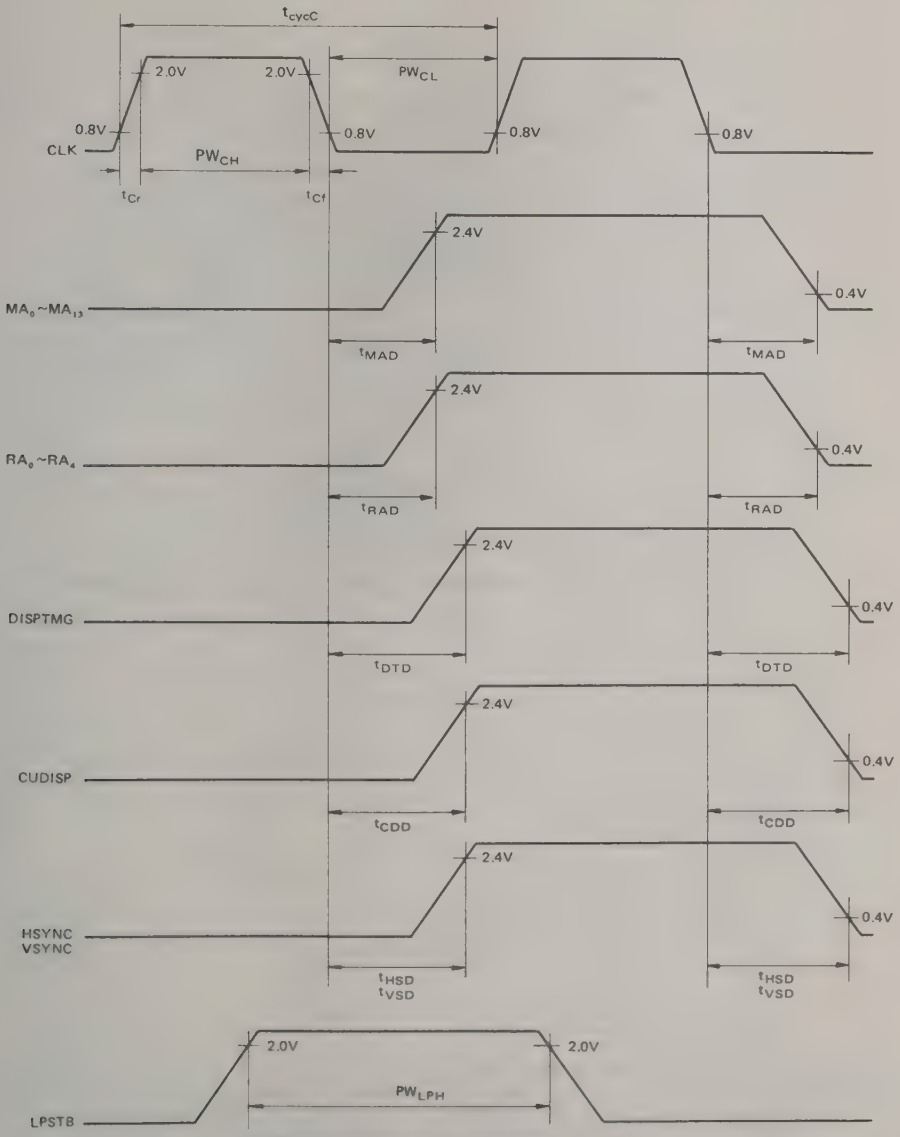
Item	Symbol	Test Condition	min	typ	max	Unit
Clock Cycle Time	t_{cycC}	Fig. 1	270	–	–	ns
Clock "High" Pulse Width	PW_{CH}		130	–	–	ns
Clock "Low" Pulse Width	PW_{CL}		130	–	–	ns
Rise andFall Time for Clock Input	t_{Cr}, t_{Cf}		–	–	20	ns
Memory Address Delay Time	t_{MAD}		–	–	160	ns
Raster Address Delay Time	t_{RAD}		–	–	160	ns
DISPTMG Delay Time	t_{DTD}		–	–	250	ns
CUDISP Delay Time	t_{CDD}		–	–	250	ns
Horizontal Sync Delay Time	t_{HSD}		–	–	200	ns
Vertical Sync Delay Time	t_{VSD}		–	–	250	ns
Light Pen Strobe Pulse Width	PW_{LPH}		60	–	–	ns
Light Pen Strobe	t_{LPD1}	Fig. 2	–	–	70	ns
Uncertain Time of Acceptance	t_{LPD2}		–	–	0	ns

2. MPU READ TIMING

Item	Symbol	Test Condition	HD6845S			HD68A45S			HD68B45S			Unit
			min	typ	max	min	typ	max	min	typ	max	
Enable Cycle Time	t_{cycE}	Fig. 3	1.0	–	–	0.666	–	–	0.5	–	–	μs
Enable "High" Pulse Width	PW_{EH}		0.45	–	–	0.28	–	–	0.22	–	–	μs
Enable "Low" Pulse Width	PW_{EL}		0.40	–	–	0.28	–	–	0.21	–	–	μs
Enable Rise and Fall Time	t_{Er}, t_{Ef}		–	–	25	–	–	25	–	–	25	ns
Address Set Up Time	t_{AS}		140	–	–	140	–	–	70	–	–	ns
Data Delay Time	t_{DDR}		–	–	320	–	–	220	–	–	180	ns
Data Hold Time	t_H		10	–	–	10	–	–	10	–	–	ns
Address Hold Time	t_{AH}		10	–	–	10	–	–	10	–	–	ns
Data Access Time	t_{ACC}		–	–	460	–	–	360	–	–	250	ns

3. MPU WRITE TIMING

Item	Symbol	Test Condition	HD6845S			HD68A45S			HD68B45S			Unit
			min	typ	max	min	typ	max	min	typ	max	
Enable Cycle Time	t_{cycE}	Fig. 4	1.0	–	–	0.666	–	–	0.5	–	–	μs
Enable "High" Pulse Width	PW_{EH}		0.45	–	–	0.28	–	–	0.22	–	–	μs
Enable "Low" Pulse Width	PW_{EL}		0.40	–	–	0.28	–	–	0.21	–	–	μs
Enable Rise and Fall Time	t_{Er}, t_{Ef}		–	–	25	–	–	25	–	–	25	ns
Address Set Up Time	t_{AS}		140	–	–	140	–	–	70	–	–	ns
Data Set Up Time	t_{DSW}		195	–	–	80	–	–	60	–	–	ns
Data Hold Time	t_H		10	–	–	10	–	–	10	–	–	ns
Address Hold Time	t_{AH}		10	–	–	10	–	–	10	–	–	ns



This Figure shows the relation in time between CLK signal and each output signals. Output sequence is shown in Figs. 10~15.

Figure 1 Time Chart of the CRTC

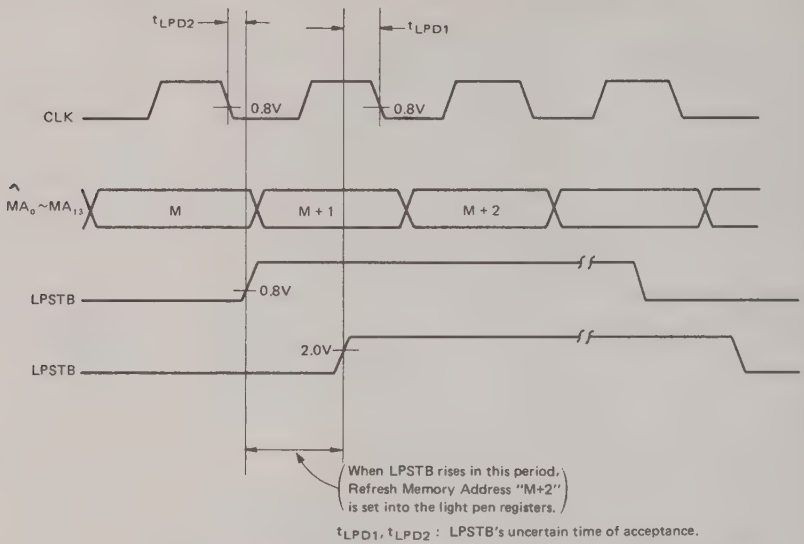


Figure 2 LPSTB Input Timing & Refresh Memory Address that is set into the light pen registers.

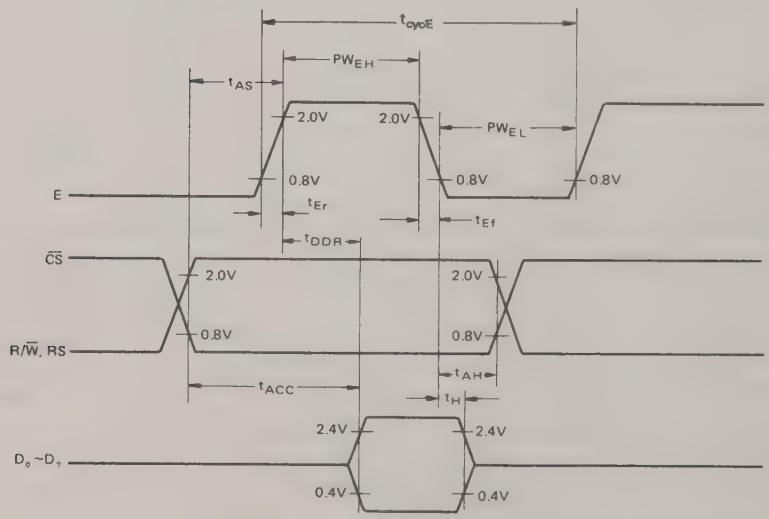


Figure 3 Read Sequence

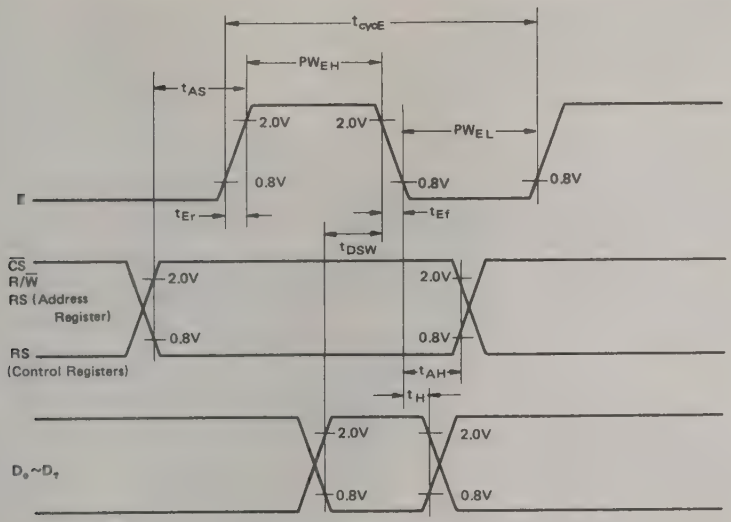
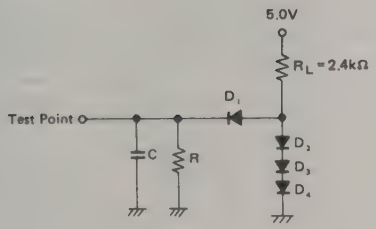


Figure 4 Write Sequence



C = 130pF (D_0~D_7)
= 30pF (Output signals except D_0~D_7)
R = 11kΩ (D_0~D_7)
= 24kΩ (Output signals except D_0~D_7)
D_1~D_4 are 1S2074Ⓗ or equivalent.

Figure 5 Test Loads

■ SYSTEM DESCRIPTION

The CRTC is a LSI which is connected with MPU and CRT display device to control CRT display. The CRTC consists of internal register group, horizontal and vertical timing circuits, linear address generator, cursor control circuit, and light pen detection circuit. Horizontal and vertical timing circuit generate RA_0~RA_4, DISPTMG, HSYNC, and VSYNC. RA_0~RA_4 are raster address signals and used as input signals for Character Generator. DISPTMG, HSYNC, and VSYNC signals are received by video control circuit. This horizontal and vertical timing circuit consists of internal counter and comparator circuit. Linear address generator generates refresh memory address MA_0 ~MA_{13} to be used for refreshing the screen. By these address signals, refresh memory is accessed periodically. As 14 refresh memory address signals are prepared, 16k words max are accessible. Moreover, the use of start address register enables paging and scrolling. Light pen detection circuit detects light pen position on the screen. When light pen strobe signal is received, light pen register memorizes linear address generated by linear address generator in order to memorize where light pen is on the screen. Cursor control circuit controls the position of cursor, its height, and its blink.

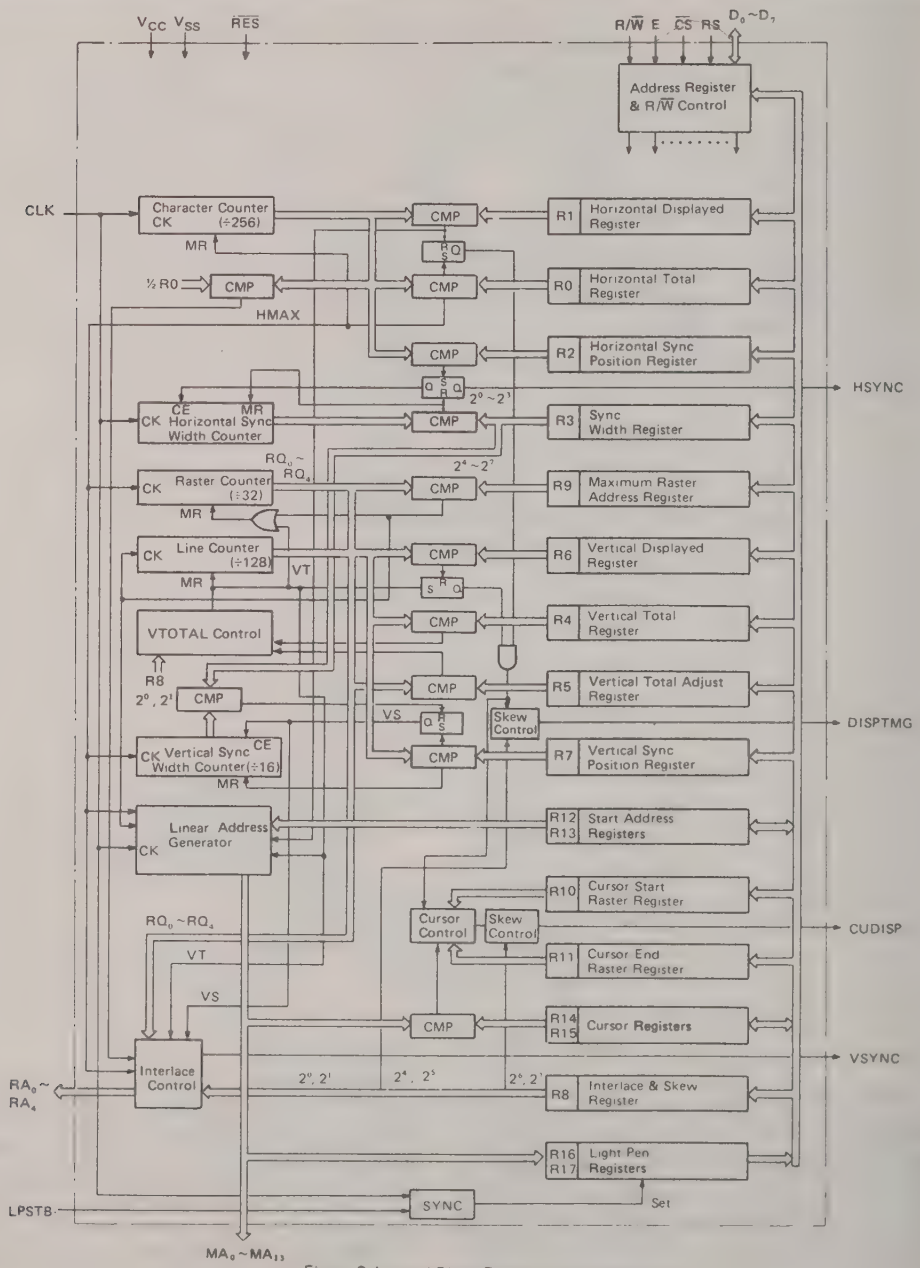


Figure 6 Internal Block Diagram of the CRTC

■ FUNCTION OF SIGNAL LINE

The CRTC provides 13 interface signals to MPU and 25 interface signals to CRT display.

● Interface Signals to MPU

Bi-directional Data Bus (D_0~D_7)

Bi-directional data bus(D_0~D_7) are used for data transfer between the CRTC and MPU. The data bus outputs are 3-state buffers and remain in the high-impedance state except when MPU performs a CRTC read operation.

Read/Write (R/$\overline{W}$)

Read/Write signal (R/$\overline{W}$) controls the direction of data transfer between the CRTC and MPU. When R/$\overline{W}$ is at "High" level, data of CRTC is transfered to MPU. When R/$\overline{W}$ is at "Low" level, data of MPU is transfered to CRTC.

Chip Select ($\overline{CS}$)

Chip Select signal ($\overline{CS}$) is used to address the CRTC. When $\overline{CS}$ is at "Low" level, it enables Read/Write operation to CRTC internal registers. Normally this signal is derived from decoded address signal of MPU under the condition that VMA of MPU is at "High" level.

Register Select (RS)

Register Select signal (RS) is used to select the address register and 18 control registers of the CRTC. When RS is at "Low" level, the address register is selected and when RS is at "High" level, control registers are selected. This signal is normally a derivative of the lowest bit (A0) of MPU address bus.

Enable(E)

Enable signal (E) is used as strobe signal in MPU Read/Write operation with the CRTC internal registers. This signal is normally a derivative of the HMCS6800 System ϕ_2 clock.

Reset ($\overline{RES}$)

Reset signal ($\overline{RES}$) is an input signal used to reset the CRTC. When $\overline{RES}$ is at "Low" level, it forces the CRTC into the following status.

1) All the counters in the CRTC are cleared and the device stops the display operation.
2) All the outputs go down to "Low" level.
3) Control registers in the CRTC are not affected and remain unchanged.

This signal is different from other HMCS6800 family LSIs in the following functions and has restrictions for usage.

1) $\overline{RES}$ has capability of reset function only when LPSTB is at "Low" level.
2) The CRTC starts the display operation immediately after $\overline{RES}$ goes "High" level.

● Interface Signals to CRT Display Device

Character Clock (CLK)

CLK is a standard clock input signal which defines character timing for the CRTC display operation. CLK is normally derived from the external high-speed dot timing logic.

Horizontal Sync (HSYNC)

HSYNC is an active "High" level signal which provides horizontal synchronization for display device.

Vertical Sync (VSYNC)

VSYNC is an active "High" level signal which provides vertical synchronization for display device.

Display Timing (DISPTMG)

DISPTMG is an active "High" level signal which defines the display period in horizontal and vertical raster scanning. It is necessary to enable video signal only when DISPTMG is at "High" level.

Refresh Memory Address (MA_0~MA_{13})

MA_0~MA_{13} are refresh memory address signals which are used to access to refresh memory in order to refresh the CRT screen periodically. These outputs enables 16k words max. refresh memory access. So, for instance, these are applicable up to 2000 characters/screen and 8-page system.

Raster Address (RA_0~RA_4)

RA_0~RA_4 are raster address signals which are used to select the raster of the character generator or graphic pattern generator etc.

Cursor Display (CUDISP)

CUDISP is an active "High" level video signal which is used to display the cursor on the CRT screen. This output is inhibited while DISPTMG is at "Low" level. Normally this output is mixed with video signal and provided to the CRT display device.

Light Pen Strobe (LPSTB)

LPSTB is an active "High" level input signal which accepts strobe pulse detected by the light pen and control circuit. When this signal is activated, the refresh memory address (MA_0~MA_{13}) which are shown in Fig. 2 are stored in the 14-bit light pen register. The stored refresh memory address need to be corrected in software, taking the delay time of the display device, light pen, and light pen control circuits into account.

■ REGISTER DESCRIPTION

Table 1 Internal Registers Assignment

$\overline{CS}$	RS	Address Register 4 3 2 1 0	Register #	Register Name	Program Unit	READ	WRITE	Data Bit 7	6	5	4	3	2	1	0
1	×	× × × × ×			–	–	–								
0	0	× × × × ×	AR	Address Register	–	×	○								
0	1	0 0 0 0 0	R0	Horizontal Total *	Character	×	○								
0	1	0 0 0 0 1	R1	Horizontal Displayed	Character	×	○								
0	1	0 0 0 1 0	R2	Horizontal Sync* Position	Character	×	○								
0	1	0 0 0 1 1	R3	Sync Width	Vertical-Raster, Horizontal-Character	×	○	wv3	wv2	wv1	wv0	wh3	wh2	wh1	wh0
0	1	0 0 1 0 0	R4	Vertical Total *	Line	×	○								
0	1	0 0 1 0 1	R5	Vertical Total Adjust	Raster	×	○								
0	1	0 0 1 1 0	R6	Vertical Displayed	Line	×	○								
0	1	0 0 1 1 1	R7	Vertical Sync * Position	Line	×	○								
0	1	0 1 0 0 0	R8	Interlace & Skew	–	×	○	C1	C0	D1	D0			V	S
0	1	0 1 0 0 1	R9	Maximum Raster Address	Raster	×	○								
0	1	0 1 0 1 0	R10	Cursor Start Raster	Raster	×	○		B	P					
0	1	0 1 0 1 1	R11	Cursor End Raster	Raster	×	○								
0	1	0 1 1 0 0	R12	Start Address(H)	–	○	○								
0	1	0 1 1 0 1	R13	Start Address(L)	–	○	○								
0	1	0 1 1 1 0	R14	Cursor(H)	–	○	○								
0	1	0 1 1 1 1	R15	Cursor (L)	–	○	○								
0	1	1 0 0 0 0	R16	Light Pen(H)	–	○	×								
0	1	1 0 0 0 1	R17	Light Pen(L)	–	○	×								

[NOTE] 1. The Registers marked *: (Written Value) = (Specified Value) – 1
2. Written Value of R9 is mentioned below.
   1) Non-interlace Mode / Interlace Sync Mode: (Written Value) = (Specified Value) - 1
   2) Interlace Sync & Video Mode: (Written Value) = (Specified Value) -2
3. C0 and C1 specify skew of CUDISP.
   D0 and D1 specify skew of DISPTMG.
   When S is "1", V specifies video mode. S specifies the Interlace Sync Mode.
4. B specifies the cursor blink. P specifies the cursor blink period.
5. wv0~wv3 specify the pulse width of Vertical Sync Signal.
   wh0~wh3 specify the pulse width of Horizontal Sync Signal.
6. R0 is ordinally programmed to be odd number in interlace mode.
7. ○; Yes, ×; No

■ FUNCTION OF INTERNAL REGISTERS

• Address Register (AR)

This is a 5-bit register used to select 18 internal control registers (R0~R17). Its contents are the address of one of 18 internal control registers. Programming the data from 18 to 31 produces no results. Access to R0~R17 requires, first of all, to write the address of corresponding control register into this register. When RS and $\overline{CS}$ are at "Low" level, this register is selected.

• Horizontal Total Register (R0)

This is a register used to program total number of horizontal characters per line including the retrace period. The data is 8-bit and its value should be programmed according to the specification of the CRT. When M is total number of characters, M-1 shall be programmed to this register. When programming for interlace mode, M must be even.

• Horizontal Displayed Register (R1)

This is a register used to program the number of horizontal displayed characters per line. Data is 8-bit and any number that is smaller than that of horizontal total characters can be programmed.

• Horizontal Sync Position Register (R2)

This is a register used to program horizontal sync position as multiples of the character clock period. Data is 8-bit and any number that is lower than the horizontal total number can be programmed. When H is character number of horizontal Sync Position, H-1 shall be programmed to this register. When programmed value of this register is increased, the display position on the CRT screen is shifted to the left. When programmed value is decreased, the position is shifted to the right. Therefore, the optimum horizontal position can be determined by this value.

• Sync Width Register (R3)

This is a register used to program the horizontal sync pulse width and the vertical sync pulse width. The horizontal sync pulse width is programmed in the lower 4-bit as multiples of the character clock period. "0" can't be programmed. The vertical sync pulse width is programmed in higher 4-bit as multiples of the raster period. When "0" is programmed in higher 4-bit, 16 raster period (16H) is specified.

• Vertical Total Register (R4)

This is a register used to program total number of lines per frame including vertical retrace period. The data is within 7-bit and its value should be programmed according to the specification of the CRTC. When N is total number of lines, N-1 shall be programmed to this register.

• Vertical Total Adjust Register (R5)

This is a register used to program the optimum number to adjust total number of rasters per field. This register enables to decide the number of vertical deflection frequency more strictly.

• Vertical Displayed Register (R6)

This is a register used to program the number of displayed character rows on the CRT screen. Data is 7-bit and any number that is smaller than that of vertical total characters can be programmed.

Table 2 Pulse Width of Vertical Sync Signal

VSW				Pulse Width
2^7	2^6	2^5	2^4	
0	0	0	0	16H
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	10
1	0	1	1	11
1	1	0	0	12
1	1	0	1	13
1	1	1	0	14
1	1	1	1	15

H; Raster period

Table 3 Pulse Width of Horizontal Sync Signal

HSW				Pulse Width
2^3	2^2	2^1	2^0	
0	0	0	0	– (Note)
0	0	0	1	1 CH
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	10
1	0	1	1	11
1	1	0	0	12
1	1	0	1	13
1	1	1	0	14
1	1	1	1	15

CH; Character clock period
(Note) HSW = "0" can't be used.

- **Vertical Sync Position Register (R7)**

This is a register used to program the vertical sync position on the screen as multiples of the horizontal character line period. Data is 7-bit and any number that is equal to or less than vertical total characters can be programmed. When V is character number of vertical sync position, V-1 shall be programmed to this register. When programmed value of this register is increased, the display position is shifted up. When programmed value is decreased, the position is shifted down. Therefore, the optimum vertical position may be determined by this value.

- **Interlace and Skew Register (R8)**

This is a register used to program raster scan mode and skew (delay) of CUDISP and DISPTMG.

Raster Scan Mode Program Bit (V, S)

Raster scan mode is programmed in the V, S bit.

Table 4 Raster Scan Mode (2^1, 2^0)

V	S	Raster Scan Mode
0	0	Non-interlace Mode
1	0	
0	1	Interlace Sync Mode
1	1	Interlace Sync & Video Mode

In the non-interlace mode, the rasters of even number field and odd number field are scanned duplicatedly. In the interlace sync mode, the rasters of odd number field are scanned in the middle of even number field. Then it is controlled to display the same character pattern in two fields. In the interlace sync & video mode, the raster scan method is the same as the interlace sync mode, but it is controlled to display different character pattern in two field.

Skew Program Bit (C1, C0, D1, D0)

These are used to program the skew (delay) of CUDISP and DISPTMG.

Skew of these two kinds of signals are programmed separately.

Table 5 DISPTMG Skew Bit (2^5, 2^4)

D1	D0	DISPTMG
0	0	Non-skew
0	1	One-character skew
1	0	Two-character skew
1	1	Non-output

Table 6 CUDISP Skew Bit (2^7, 2^6)

C1	C0	CUDISP
0	0	Non-skew
0	1	One-character skew
1	0	Two-character skew
1	1	Non-output

Skew function is used to delay the output timing of CUDISP and DISPTMG in LSI for the time to access refresh memory, character generator or pattern generator, and to make the same phase with serial video signal.

- **Maximum Raster Address Register (R9)**

This is a register used to program maximum raster address within 5-bit. This register defines total number of rasters per character including line space. This register is programmed as follows.

Non-interlace Mode, Interlace Sync Mode

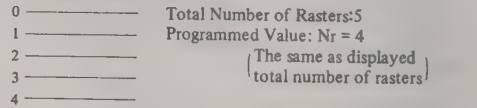
When total number of rasters is RN, RN-1 shall be programmed.

Interlace Sync & Video Mode

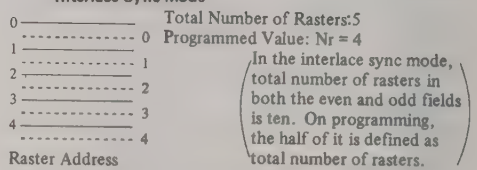
When total number of rasters is RN, RN-2 shall be programmed.

This manual defines total number of rasters in non-interlace mode, interlace sync mode and interlace sync & video mode as follows:

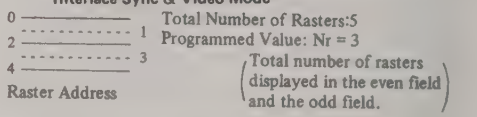
Non-interlace Mode



Interlace Sync Mode



Interlace Sync & Video Mode

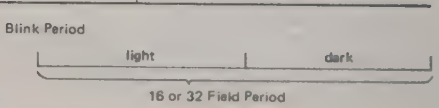


- **Cursor Start Raster Register (R10)**

This is a register used to program the cursor start raster address by lower 5-bit (2^0~2^4) and the cursor display mode by higher 2-bit (2^5, 2^6).

Table 7 Cursor Display Mode (2^6, 2^5)

B	P	Cursor Display Mode
0	0	Non-blink
0	1	Cursor Non-display
1	0	Blink 16 Field Period
1	1	Blink 32 Field Period



- **Cursor End Raster Register (R11)**
  This is register used to program the cursor end raster address.

- **Start Address Register (R12, R13)**
  These are used to program the first address of refresh memory to read out.
  Paging and scrolling is easily performed using this register. This register can be read but the higher 2-bit (2^6, 2^7) of R12 are always "0".

- **Cursor Register (R14, R15)**
  These two read/write registers stores the cursor location. The higher 2-bit (2^6, 2^7) of R14 are always "0".

- **Light Pen Register (R16, R17)**
  These read only registers are used to catch the detection address of the light pen. The higher 2-bit (2^6, 2^7) of R16 are always "0". Its value needs to be corrected by software because there is time delay from address output of the CRTC to signal input LPSTB pin of the CRTC in the process that raster is lit after address output and light pen detects it. Moreover, delay time shown in Fig. 2 needs to be taken into account.

Restriction on Programming Internal Register

1) $0 < Nhd < Nht + 1 \leqq 256$
2) $0 < Nvd < Nvt + 1 \leqq 128$
3) $0 \leqq Nhsp \leqq Nht$
4) $0 \leqq Nvsp \leqq Nvt$*
5) $0 \leqq NCSTART \leqq NCEND \leqq Nr$ (Non-interlace, Interlace sync mode)
   $0 \leqq NCSTART \leqq NCEND \leqq Nr + 1$ (Interlace sync & video mode)
6) $2 \leqq Nr \leqq 30$ (Interlace Sync & Video mode)
7) $3 \leqq Nht$ (Except non-interlace mode)
   $5 \leqq Nht$ (Non-interlace mode only)

* In the interlace mode, pulse width is changed ±½ raster time when vertical sync signal extends over two fields.

Notes for Use

The method of directly using the value programmed in the internal registers of LSI for controlling the CRT is adopted. Consequently, the display may flicker on the screen when the contents of the registers are changed from bus side asynchronously with the display operation.

Cursor Register

Writing into this register at frequent intervals for moving the cursor should be performed during horizontal and vertical retrace period.

Start Address Register

Writing into the start address register at frequent intervals for scrolling and paging should be performed during horizontal and vertical display period.

It is desirable to avoid programming other registers during display operation.

■ OPERATION OF THE CRTC

- **Time Chart of CRT Interface Signals**
  The following example shows the display operation in which values of Table 8 are programmed to the CRTC internal registers. Fig. 7 shows the CRT screen format. Fig. 10 shows the time chart of signals output from the CRTC.

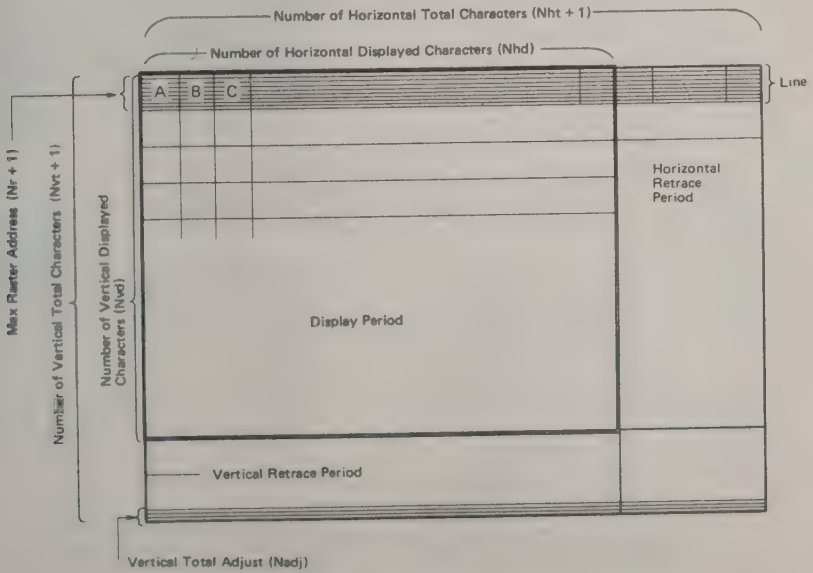


Figure 7 CRT Screen Format

Table 8 Programmed Values into the Registers

Register	Register Name	Value	Register	Register Name	Value
R0	Horizontal Total	Nht	R9	Max Raster Address	Nr
R1	Horizontal Displayed	Nhd	R10	Cursor Start Raster	
R2	Horizontal Sync Position	Nhsp	R11	Cursor End Raster	
R3	Sync Width	Nvsw, Nhsw	R12	Start Address (H)	0
R4	Vertical Total	Nvt	R13	Start Address (L)	0
R5	Vertical Total Adjust	Nadj	R14	Cursor (H)	
R6	Vertical Displayed	Nvd	R15	Cursor (L)	
R7	Vertical Sync Position	Nvsp	R16	Light Pen (H)	
R8	Interlace & Skew		R17	Light Pen (L)	

[NOTE] Nhd<Nht, Nvd<Nvt

The relation between values of Refresh Memory Address ($MA_0 \sim MA_{13}$) and Raster Address ($RA_0 \sim RA_4$) and the display position on the screen is shown in Fig. 16. Fig. 16 shows the case where the value of Start Address is 0.

- **Interlace Control**

Fig. 8 shows an example where the same character is displayed in the non-interlace mode, interlace sync mode, and interlace sync & video mode.

Non-interlace Mode Control

In non-interlace mode, each field is scanned duplicatedly. The values of raster addresses ($RA_0 \sim RA_4$) are counted up one from 0.

Interlace Sync Mode Control

In the interlace sync mode, raster addressed in the even field and the odd field are the same as addressed in the non-interlace mode. One character pattern is displayed mutually and its displayed position in the odd field is set at 1/2 raster space down from that in the even field.

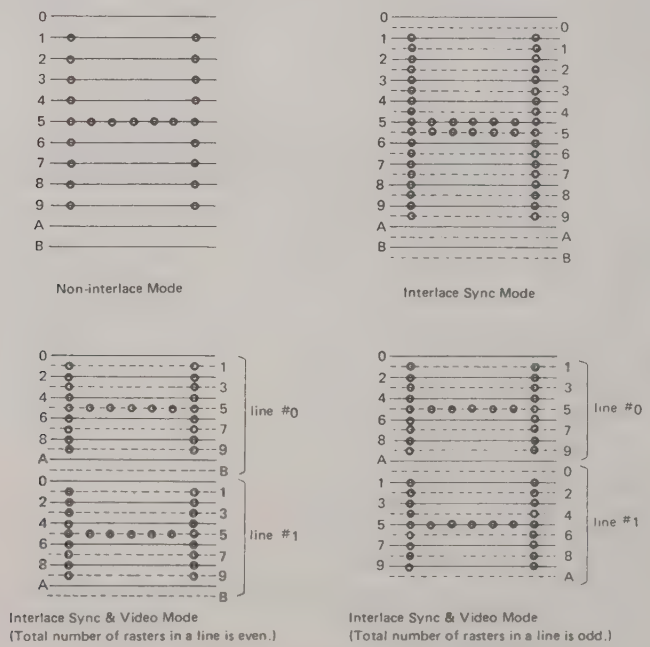

Non-interlace Mode
Interlace Sync Mode
line #0
line #1
line #0
line #1
Interlace Sync & Video Mode
(Total number of rasters in a line is even.)
Interlace Sync & Video Mode
(Total number of rasters in a line is odd.)

Figure 8 Example of Raster Scan Display

Interlace Sync & Video Mode Control

In interlace sync & video mode, the output raster address when the number of rasters is even is different from that when the number of rasters is odd.

Table 9 The Output of Raster Address in Interlace Sync & Video Mode

Total Number of Rasters in a Line \ Field		Even Field	Odd Field
Even		Even Address	Odd Address
Odd	Even Line*	Even Address	Odd Address
	Odd Line*	Odd Address	Even Address

* Internal line address begins from 0.

1) Total number of rasters in a line is even;

When number of rasters is programmed to be even, even raster address is output in the even field and odd raster address is output in the odd field.

2) Total number of rasters in a line is odd;

When total number of rasters is programmed to be odd, odd and even addresses are reversed according to the odd and even lines in each field. In this case, the difference in numbers of dots displayed between even field and odd field is usually smaller the case of 1). Then interlace can be displayed more stably.

[NOTE] The wide disparity of dots between number of dots between even field and odd field influences beam current of CRT. CRT, which has a stable high-voltage part, can make interlace display normal. On the contrary, CRT, which has unstable high-voltage part, moves deflection angle of beam current and also dots displayed in the even and odd fields may be shifted. Characters appears distroting on a border of the screen. So 2) programming has an effect to decrease such evil influences as mentioned above. Fig. 13 shows fine chart in each mode when interlace is performed.

- **Cursor Control**

Fig. 9 shows the display patterns where each value is programmed to the cursor start raster register and the cursor end raster register. Programmed values to the cursor start raster register and the cursor end raster register need to be under the following condition.

Cursor Start Raster Register ≦ Cursor End Raster Register ≦ Maximum Raster Address Register.

Time chart of CUDISP is shown in Fig. 14 and Fig. 15.

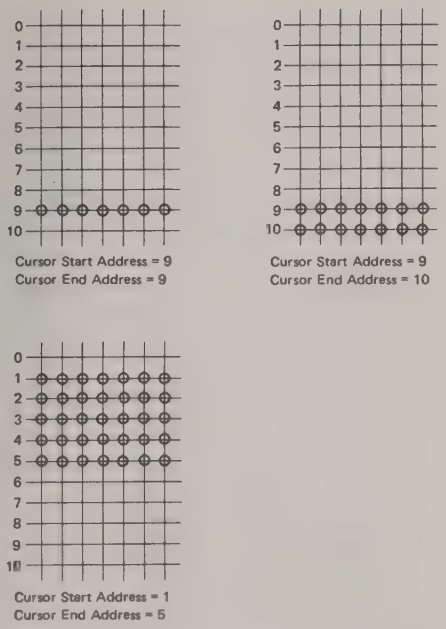


Figure 9 Cursor Control

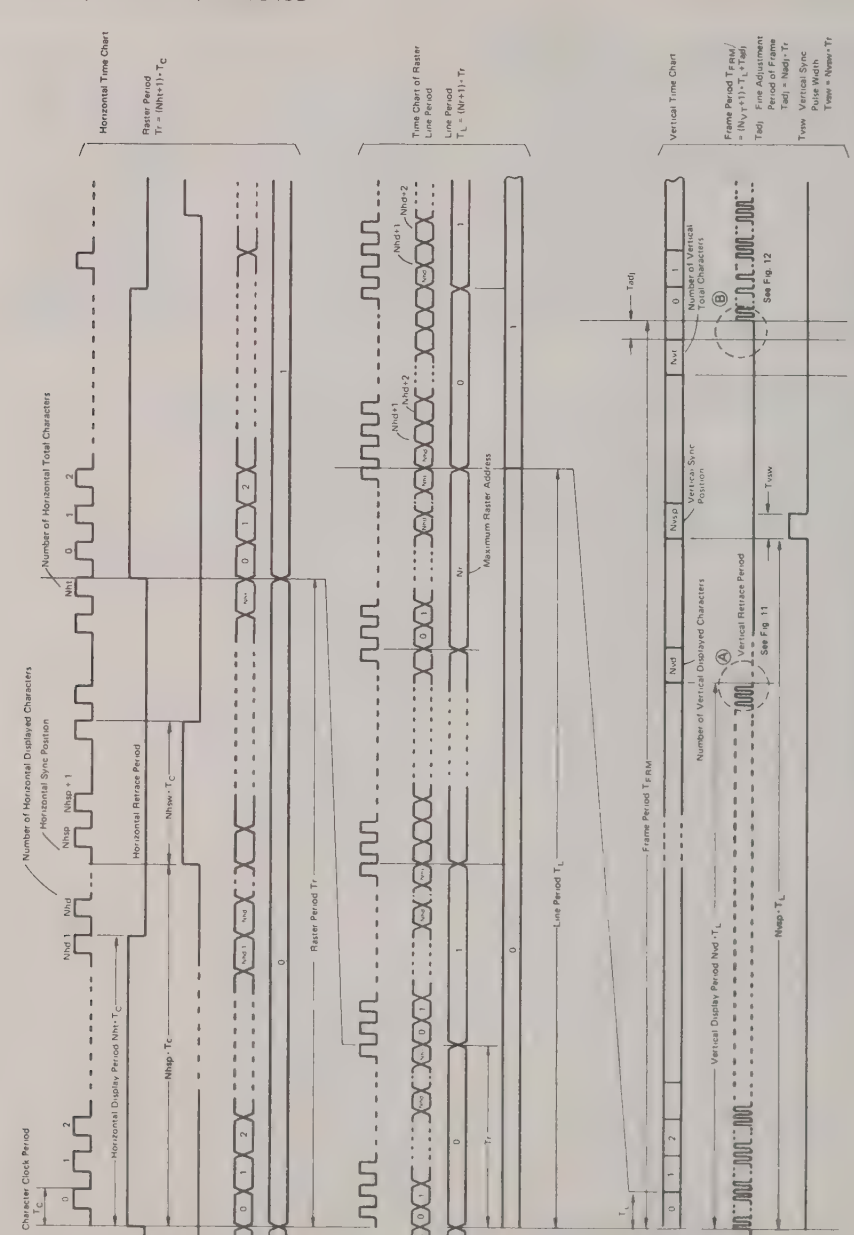


Figure 10 CRTC Time Chart

(Output waveform of horizontal & vertical display in the case where values shown in Table 8 are Programmed to each register.)

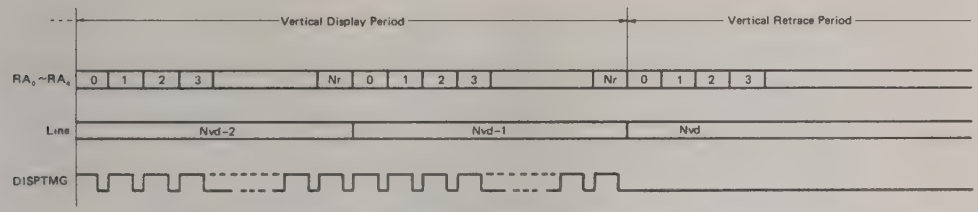


Figure 11 Switching from Vertical Display Period over to Vertical Retrace Period (Expansion of Fig. 10– Ⓐ)

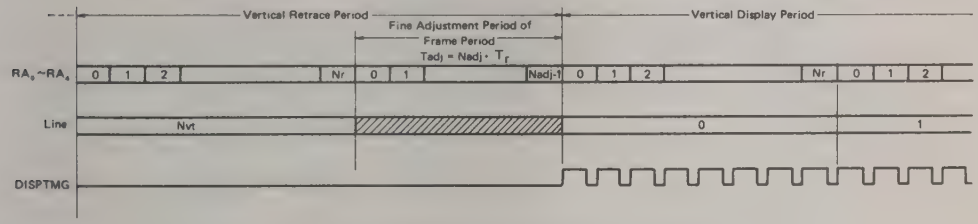


Figure 12 Fine Adjustment Period of Frame in Vertical Display
(Expansion of Fig. 10– Ⓑ)

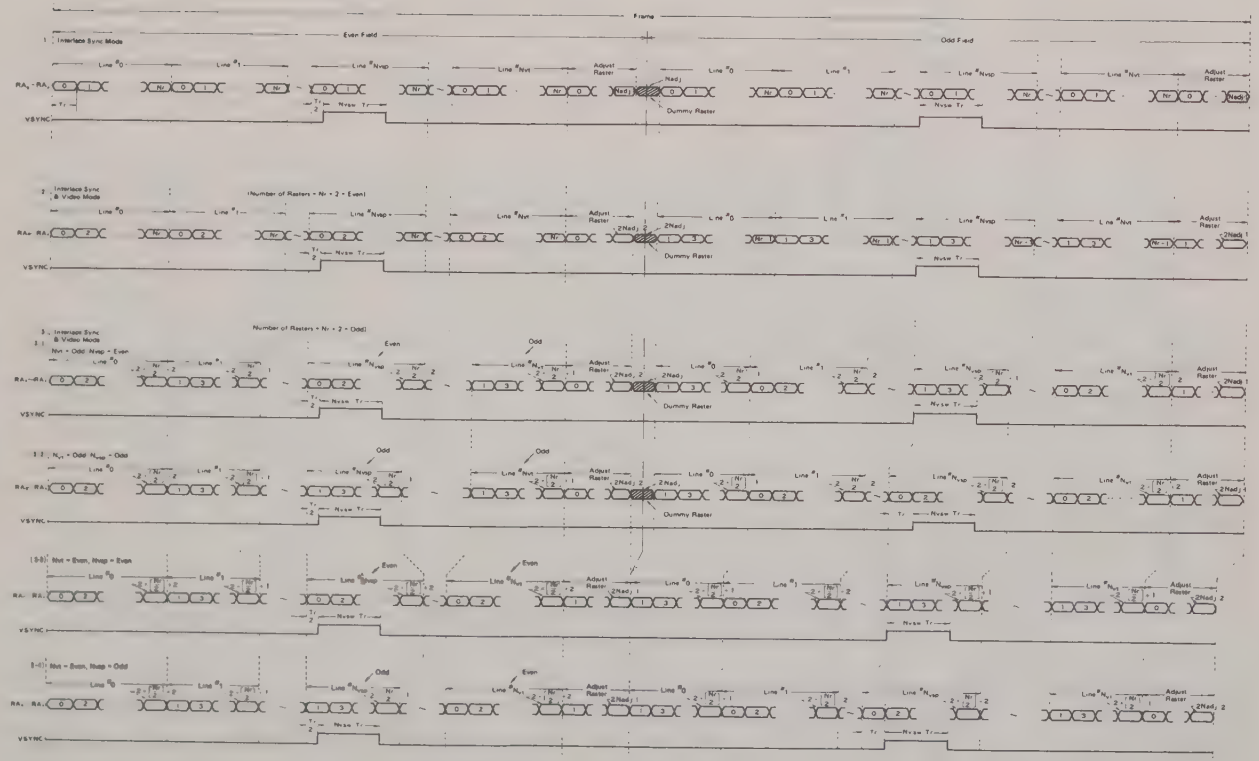

Frame
Even Field
Odd Field
Interlace Sync Mode
Dummy Raster
Interlace Sync & Video Mode
Number of Rasters = Nr + 2 = Even
Number of Rasters = Nr + 2 = Odd
Nvt = Odd, Nvsp = Even
Nvt = Odd, Nvsp = Odd
Nvt = Even, Nvsp = Even
Nvt = Even, Nvsp = Odd
VSYNC

Figure 13 Interlace Control

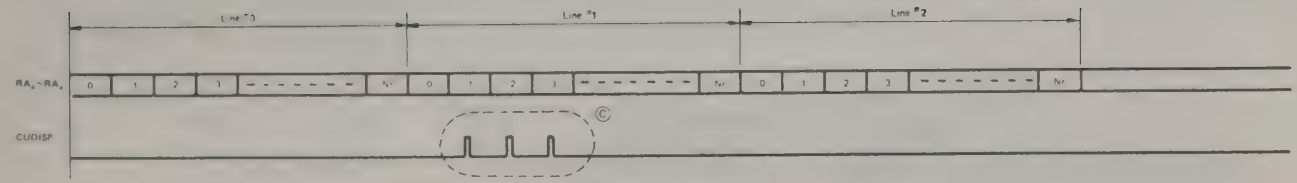
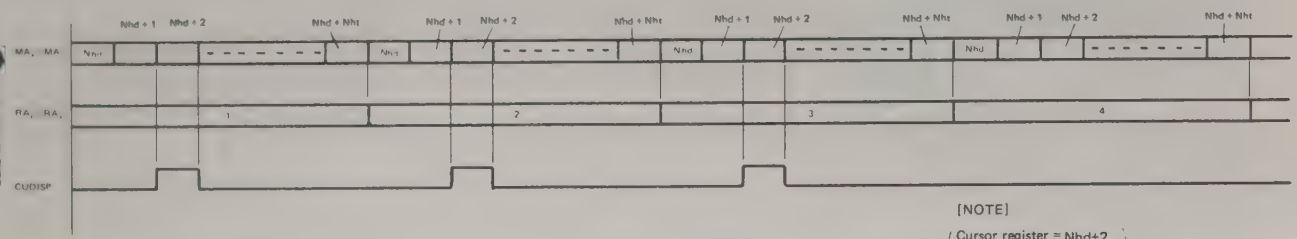


Figure 14 Relation between Line · Raster and CUDISP

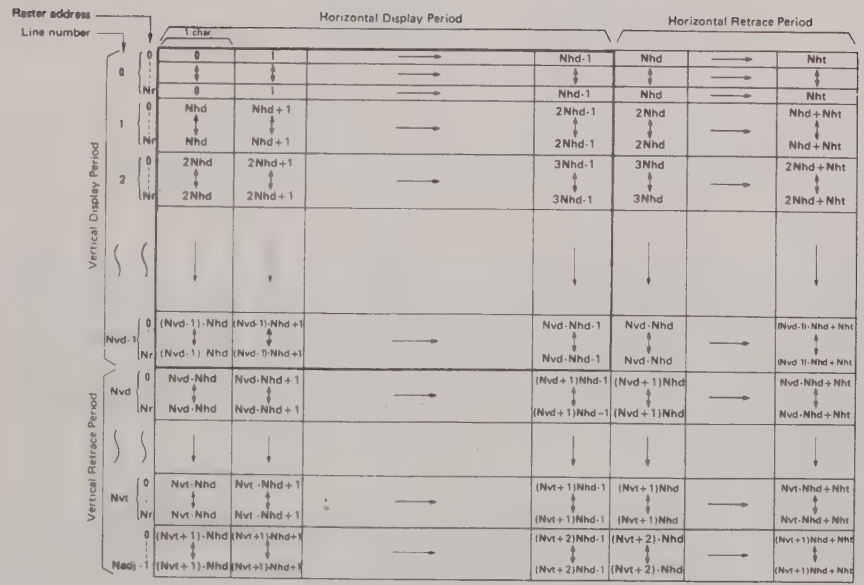


[NOTE]

(Cursor register = Nhd+2
Cursor Start
Raster Register = 1
Cursor End
Raster Register = 3)
are Programmed in cursor display mode.

In blink mode, it is changed into display or non-display mode when field period is 16 or 32-time period.

Figure 15 CUDISP Timing (Expansion of Fig. 14. ©)



Valid refresh memory address (0~Nvd·Nhd-1) are shown within the thick-line square. Refresh memory address are provided even during horizontal and Vertical retrace period. This is an example in the case where the programmed value of start address register is 0.

Figure 16 Refresh Memory Address (MA_0~MA_{13})

HITACHI

■ How to Use the CRTC

• Interface to MPU

As shown in Fig. 17, the CRTC is connected with the standard bus of MPU to control the data transfer between them. The CRTC address is determined by $\overline{CS}$ and RS, and the Read/Write operation is controlled by R/$\overline{W}$ and E. When $\overline{CS}$ is "Low" and RS is also "Low", the CRTC address register is selected. When $\overline{CS}$ is "Low" and RS is "High", one of 18 internal registers is selected.

$\overline{RES}$ is the system reset signal. When $\overline{RES}$ becomes "Low", the CRTC internal control logic is reset. But internal registers shown in Table 1 (R0~R17) are not affected by $\overline{RES}$ and remain unchanged.

The CRTC is designed so as to provide an interface to microcomputers, but adding some external circuits enables an interface to other data sources.

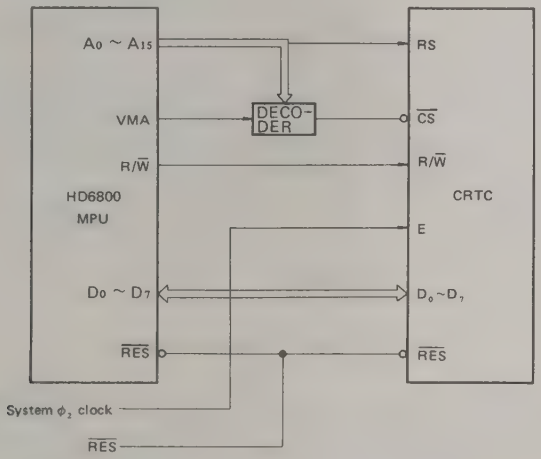


Figure 17 Interface to MPU

• Dot Timing Generating Circuit

CRTC's CLK input (21 pin) is provided with CLK which defines horizontal character time period from the outside. This CLK is generated by dot counter shown in Fig. 18. Fig. 18 shows a example of circuit where horizontal dot number of the character is "9". Fig. 19 shows the operation time chart of dot counter shown in Fig. 18. As this example shows explicitly, CLK is at "Low" level in the former half of horizontal character time and at "High" level in the latter half. It is necessary to be careful so as not to mistake this polarity.

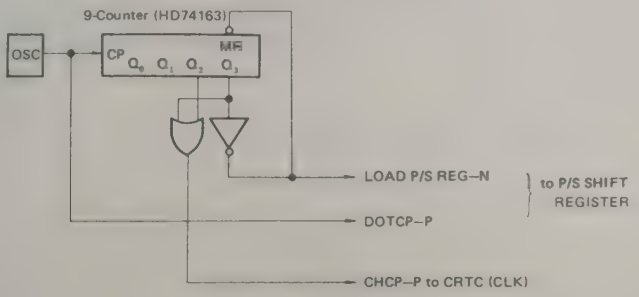


Figure 18 Example of Dot Counter

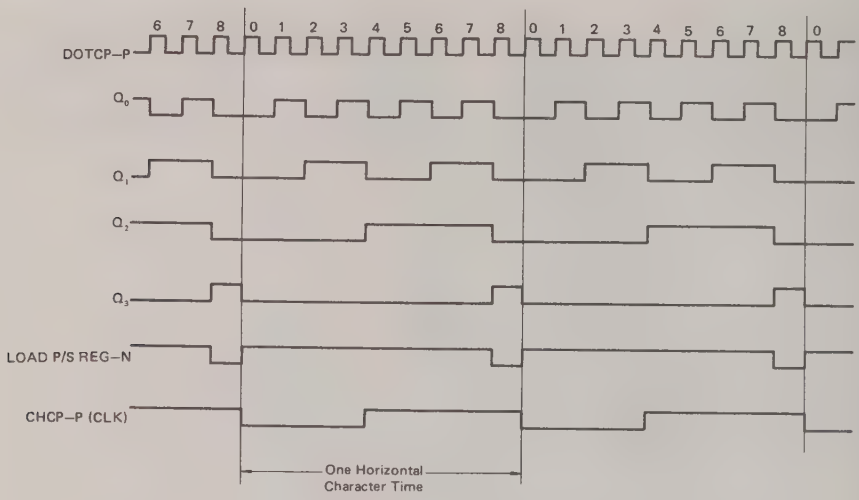


Figure 19 Time Chart of Dot Counter

■ INTERFACE TO DISPLAY CONTROL UNIT

Fig. 20 shows the interface between the CRTC and display control unit. Display control unit is mainly composed of Refresh Memory, Character Generator, and Video Control circuit. For refresh memory, 14 Memory Address line (0~16383) max are provided and for character generator, 5 Raster Address line (0~31) max are provided. For video control circuit, DISPTMG, CUDISP, HSYNC, and VSYNC are sent out. DISPTMG is used to control the blank period of video signal. CUDISP is used as video signal to display the cursor on the CRT screen. Moreover, HSYNC and VSYNC are used as drive signals respectively for CRT horizontal and vertical deflection circuits.

Outputs from video control circuit, (video signals and sync signals) are provided to CRT display unit to control the deflection and brightness of CRT, thus characters are displayed on the screen.

Fig. 21 shows detailed block diagram of display control unit. This shows how to use CUDISP and DISPTMG. CUDISP and DISPTMG should be used being latched at least one time at external flip-flop F1 and F2. Flip-flop F1 and F2 function to make one-character delay time so as to synchronize them with video signal from parallel-serial converter. High-speed D type flip-flop as TTL is used for this purpose. After being delayed at F1 and F2 DISPTMG is AND-ed with character video signal, and CUDISP is Or-ed with output from AND gate. By using this circuitry, blanking of horizontal and vertical retrace time is controlled. And cursor video is mixed with character video signal.

Fig. 21 shows the example in the case that both refresh memory and Character Generator can be accessed for horizontal one character time. Time chart for this case is shown in Fig. 24. This method is used when a few character needed to be displayed in horizontal direction on the screen.

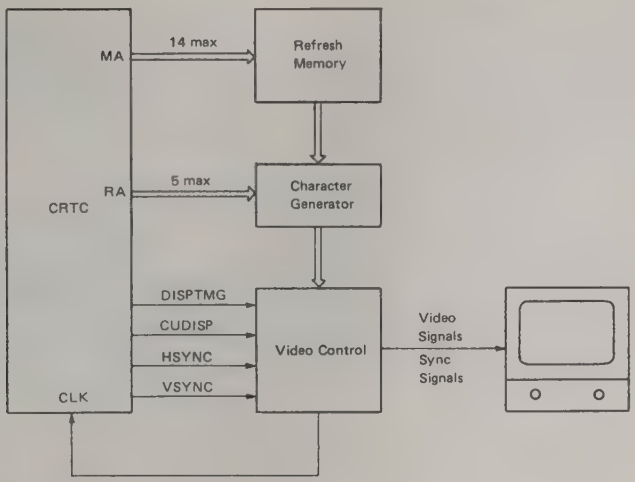


Figure 20 Interface to Display Control Unit

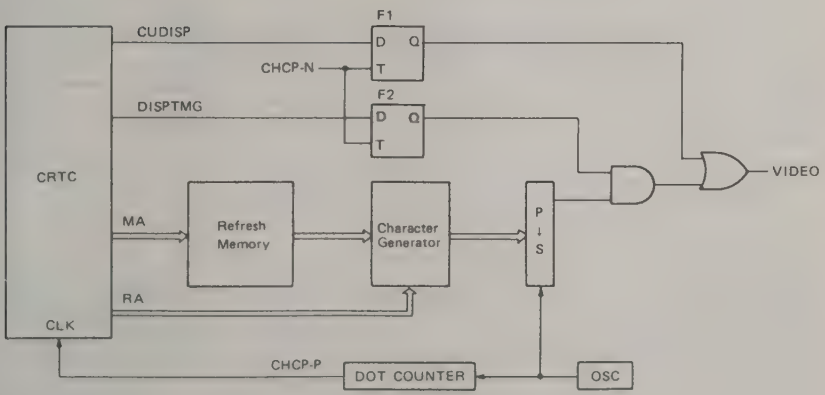


Figure 21 Display Control Unit (1)

When many characters are displayed in horizontal direction on the screen, and horizontal one-character time is so short that both refresh memory and Character Generator cannot be accessed, the circuitry shown in Fig. 22 should be used. In this case refresh memory output shall be latched and Character Generator shall be accessed at the next cycle. The time chart in this case is shown in Fig. 25. CUDISP and DISPTMG should be provided after being delayed by one-character time by using skew bit of interlace & skew register (R8). Moreover, when there are some troubles about delay time of MA during horizontal one-character time on high-speed display operation, system shown in Fig. 23 is adopted. The time chart in this case is shown in Fig. 26. Character video signal is delayed for two-character time because each MA outputs and refresh memory outputs are latched, and they are made to be in phase with CUDISP and DISPTMG by delaying for two-character time. Table 10 shows the circuitry selection standard of display units.

Table 10 Circuitry Standard of Display Control Unit

Case	Relation among t_{CH} Refresh Memory and Character Generator	Block Diagram	Interlace & Skew Register Bit Programming			
			C1	C0	D1	D0
1	$t_{CH} >$ RM Access + CG Access + t_{MAD}	Fig. 21	0	0	0	0
2	RM Access + CG Access + $t_{MAD} \geqq t_{CH} >$ RM Access + t_{MAD}	Fig. 22	0	1	0	1
3	RM Access + $t_{MAD} \geqq t_{CH} >$ RM Access	Fig. 23	1	0	1	0

t_{CH} : CHCP Period; t_{MAD} : MA Delay
RM: Refresh Memory CG: Character Generator

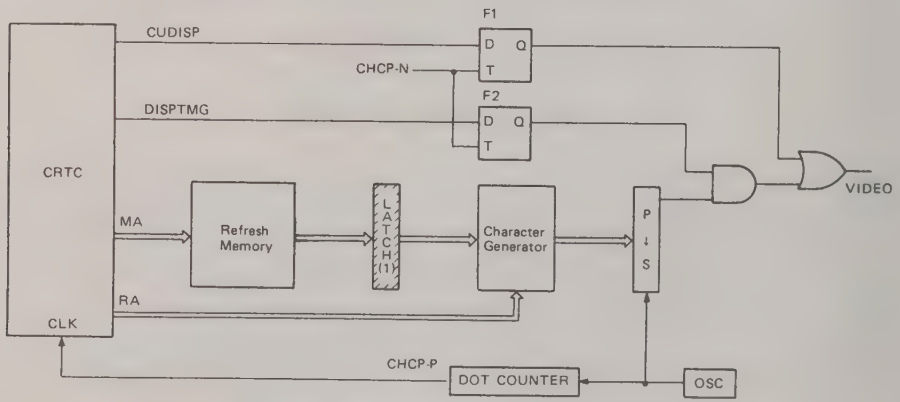


Figure 22 Display Control Unit (2)

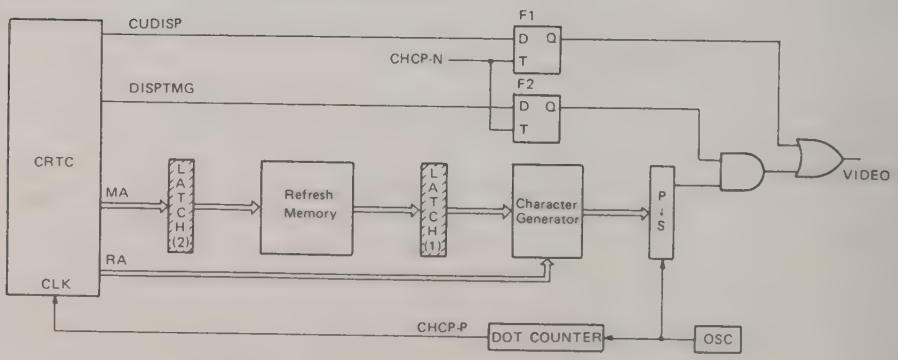


Figure 23 Display Control Unit (For high-speed display operation) (3)

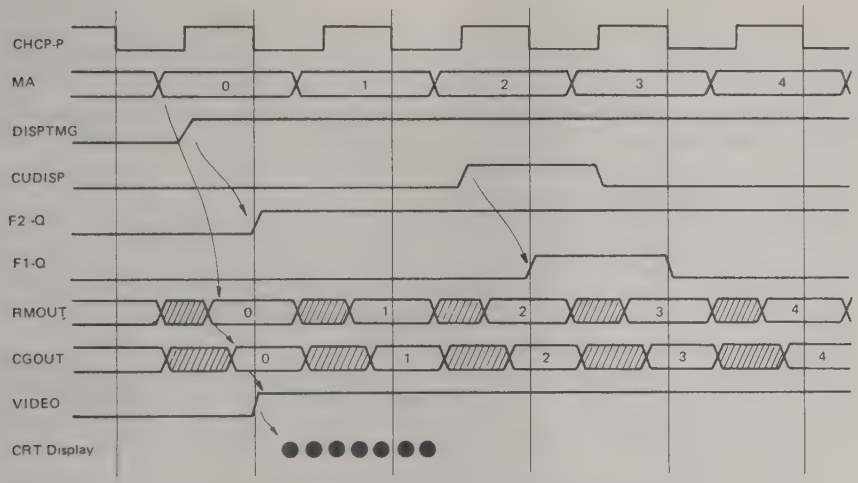


Figure 24 Time Chart of Display Control Unit (1)

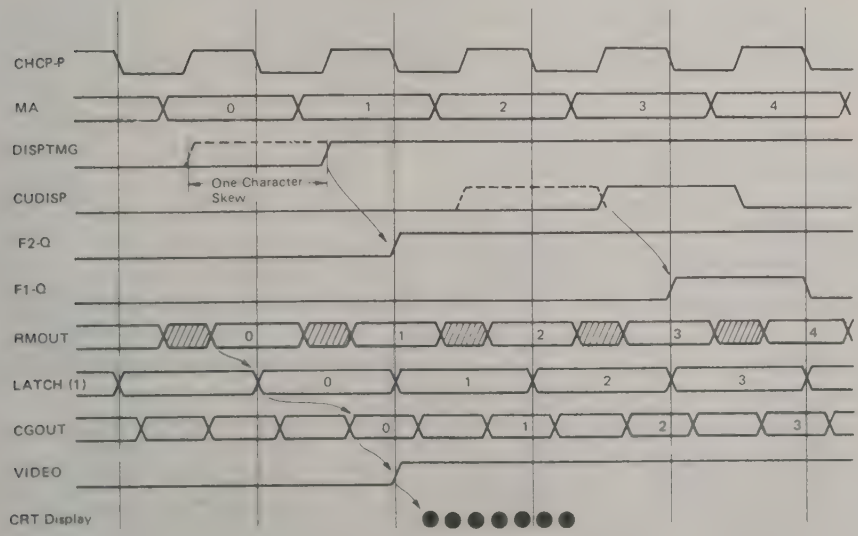


Figure 25 Time Chart of Display Control Unit (2)

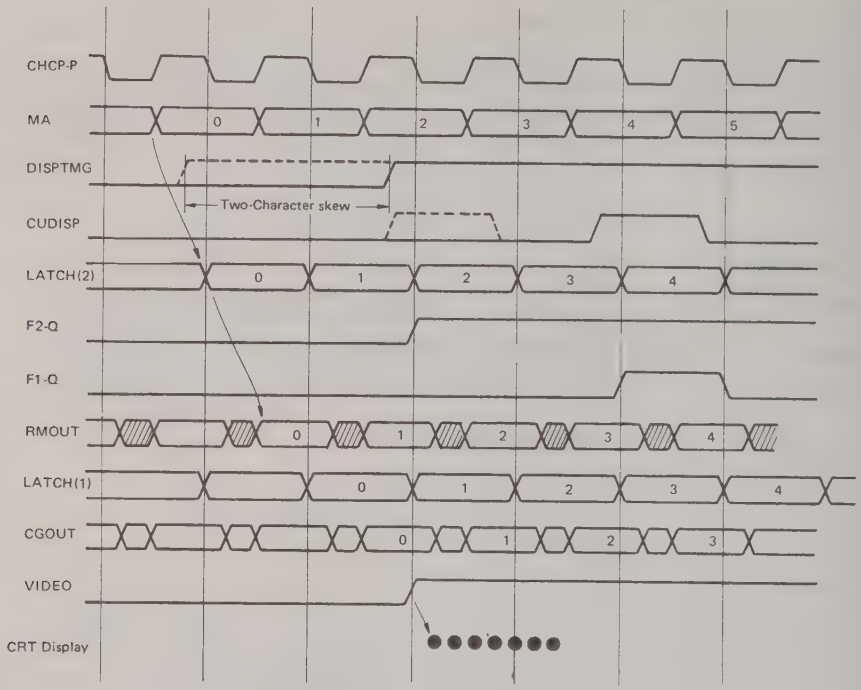


Figure 26 Time Chart of Display Unit (3)

■ HOW TO DECIDE PARAMETERS SET ON THE CRTC

● How to Decide Parameters Based on Specification of CRT Display Unit (Monitor)

Number of Horizontal Total Characters

Horizontal deflection frequency f_h is given by specification of CRT display unit. Number of horizontal total characters is determined by the following equation.

$$f_h = \frac{1}{t_C\,(N_{ht} + 1)}$$

where,

t_C : Cycle Time of CLK (Character Clock)
N_{ht} : Programmed Value of Horizontal Total Register (R0)

Number of Vertical Total Characters

Vertical deflection frequency is given by specification of CRT display unit. Number of vertical Total characters is determined by the following equation.

1) Non-interlace Mode
$R_t = (N_{vt} + 1)\,(N_r + 1) + N_{adj}$
2) Interlace Sync Mode
$R_t = (N_{vt} + 1)\,(N_r + 1) + N_{adj} + 0.5$
3) Interlace Sync & Video Mode

$$R_t = \frac{(N_{vt} + 1)\,(N_r + 2) + 2N_{adj}}{2} \quad \cdots\cdots\cdots\cdots \text{(a)}$$

$$R_t = \frac{(N_{vt} + 1)\,(N_r + 2) + 2N_{adj} + 1}{2} \quad \cdots\cdots\cdots\cdots \text{(b)}$$

(a) is applied when both total numbers of vertical characters ($N_{vt} + 1$) and that of rasters in a line ($N_r + 2$) are odd.
(b) is applied when total number of rasters ($N_r + 2$) is even, or when ($N_r + 2$) is odd and total number of vertical characters ($N_{vt} + 1$) is even.

where,

R_t : Number of Total Rasters per frame (Including retrace period)
N_{vt} : Programmed Value of Vertical Total Register (R4)
N_r : Programmed Value of Maximum Raster Address Register (R9)
N_{adj} : Programmed Value of Vertical Total Adjust Register (R5)

Horizontal Sync Pulse Width

Horizontal sync pulse width is programmed to low order 4-bit of horizontal sync width register (R3) in unit of horizontal character time. Programmed value can be selected within from 1 to 15.

Horizontal Sync Position

As shown in Fig. 27, horizontal sync position is normally selected to be in the middle of horizontal retrace period. But there are some cases where is optimum sync position is not located in the middle of horizontal retrace period according to specification of CRT. Therefore, horizontal sync position should be determined by specification of CRT. Horizontal sync pulse position is programmed in unit of horizontal character time.

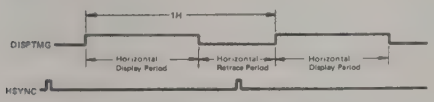


Figure 27 Time Chart of HSYNC

Vertical Sync Pulse Width

Vertical Sync Pulse Width is programmed to high order 4-bit of vertical sync pulse width register (R3) in unit of raster period. Programmed value can be selected within from 1 to 16.

Vertical Sync Position

As shown in Fig. 28, vertical sync position is normally selected to be in the middle of vertical retrace period. But there are some cases where its optimum sync position is not located in the middle of vertical retrace period according to specification of CRT. Therefore, vertical sync position should be determined by specification of CRT. Vertical sync pulse position is programmed to vertical sync position register (R7) in unit of line period.

- **How to Decide Parameters Based on Screen Format**

Dot Number of Characters (Horizontal)

Dot number of characters (horizontal) is determined by character font and character space. An example is shown in Fig. 29. More strictly, dot number of characters (horizontal) N is determined by external N-counter. Character space is set by means shown in Fig. 30.

Dot Number of Characters (Vertical)

Dot number of characters (vertical) is determined by characters font and line space. An example is shown in Fig. 29. Dot number of characters (vertical) is programmed to maximum raster address (R9) of CRTC.

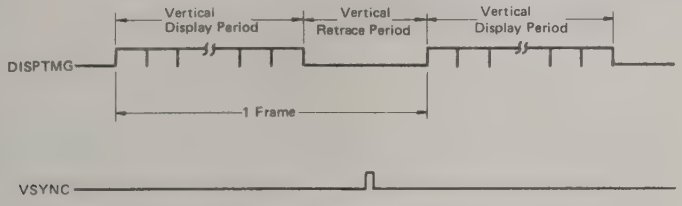


Figure 28 Time Chart of VSYNC

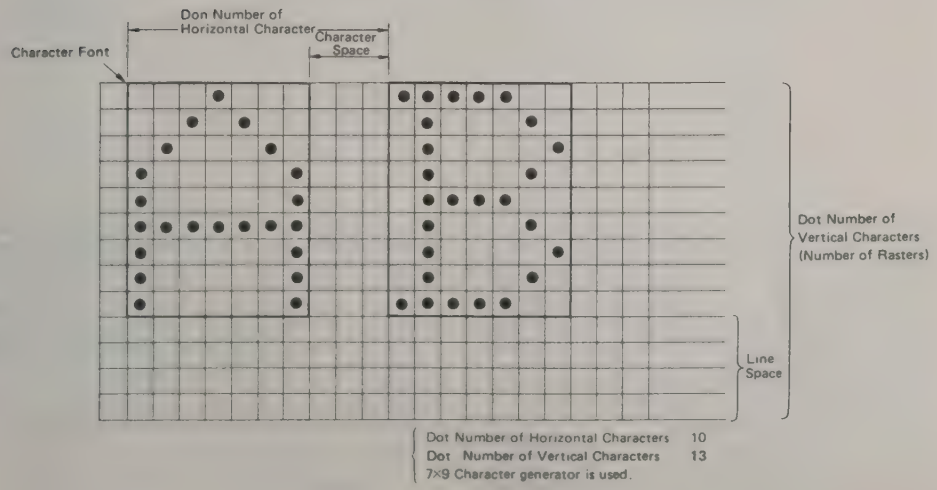

Don Number of Horizontal Character
Character Font
Character Space
Dot Number of Vertical Characters (Number of Rasters)
Line Space
Dot Number of Horizontal Characters 10
Dot Number of Vertical Characters 13
7×9 Character generator is used.

Figure 29 Dot Number of Horizontal and Vertical Characters

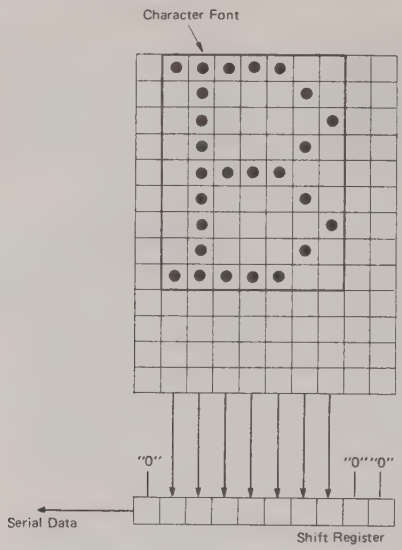


Figure 30 How to Make Character Space

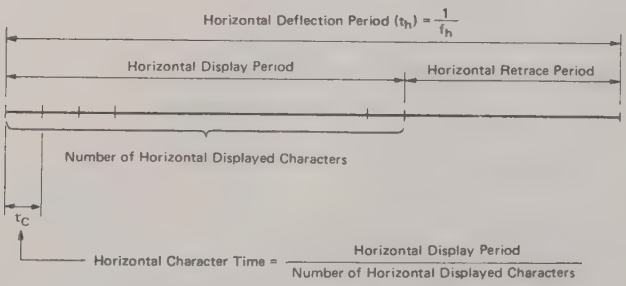

Horizontal Deflection Period (t_h) = $\frac{1}{f_h}$
Horizontal Display Period
Horizontal Retrace Period
Number of Horizontal Displayed Characters
t_C
Horizontal Character Time = $\frac{\text{Horizontal Display Period}}{\text{Number of Horizontal Displayed Characters}}$

Figure 31 Number of Horizontal Displayed Characters

Number of Horizontal Displayed Characters

Number of horizontal displayed characters is programmed to horizontal displayed register (R1) of the CRTC. Programmed value is based on screen format. Horizontal display period, which is given by specification of horizontal deflection frequency and horizontal retrace period of CRT display unit, determines horizontal character time, being divided by number of horizontal displayed characters. Moreover, its cycle time and access time which are necessary for CRT display system are determined by horizontal character time.

Number of Vertical Displayed Characters

Number of vertical displayed characters is programmed to vertical displayed register (R6). Programmed value is based on screen format. As specification of vertical deflection frequency of CRT determines number of total rasters (Rt) including vertical retrace period and the relation between number of vertical displayed character and total number of rasters on a screen is as mentioned above, CRT which is suitable for desired screen format should be selected.

For optimum screen format, it is necessary to adjust number of rasters per line, number of vertical displayed characters, and total adjust raster (Nadj) within specification of vertical deflection frequency.

Scan Mode

The CRTC can program three-scan modes shown in Table 11 to interlace mode register (R8). An example of character display in each scan mode is shown in Fig. 8.

Table 11 Program of Scan Mode

V	S	Scan Mode	Main Usage
0	0	Non-interlace	Normal Display of Characters & Figures
1	0		
0	1	Interlace Sync	Fine Display of Characters & Figures
1	1	Interlace Sync & Video	Display of Many Characters & Figures Without Using High-resolution CRT

[NOTE] In the interlace mode, the number of times per sec. in raster scanning on one spot on the screen is half as many as that in non-interlace mode. Therefore, when persistence of luminescence is short, flickering may happen. It is necessary to select optimum scan mode for the system, taking characteristics of CRT, raster scan speed, and number of displayed characters and figures into account.

Cursor Display Method

Cursor start raster register and cursor end raster register (R10, R11) enable programming the display modes shown in Table 7 and display patterns shown in Fig. 9. Therefore, it is possible to change the method of cursor display dynamically according to the system conditions as well as to realize the cursor display that meets the system requirements.

Start Address

Start address resisters (R12, R13) give an offset to the address of refresh memory to read out. This enables paging and scrolling easily.

Cursor Register

Cursor registers (R14, R15) enable programming the cursor display position on the screen. As for cursor address, it is not X, Y address but linear address that is programmed.

■ Applications of the CRTC

● Monochrome Character Display

Fig. 32 shows a system of monochrome character display. Character clock signal (CLK) is provided to the CRTC through OSC and dot counter. It is used as basic clock which drives internal control circuits. MPU is connected with the CRTC by standard bus and controls the CRTC initialization and read/write of internal registers.

Refresh memory is composed of RAM which has capacity of one frame at least and the data to be displayed is coded and stored. The data to refresh memory is changed through MPU

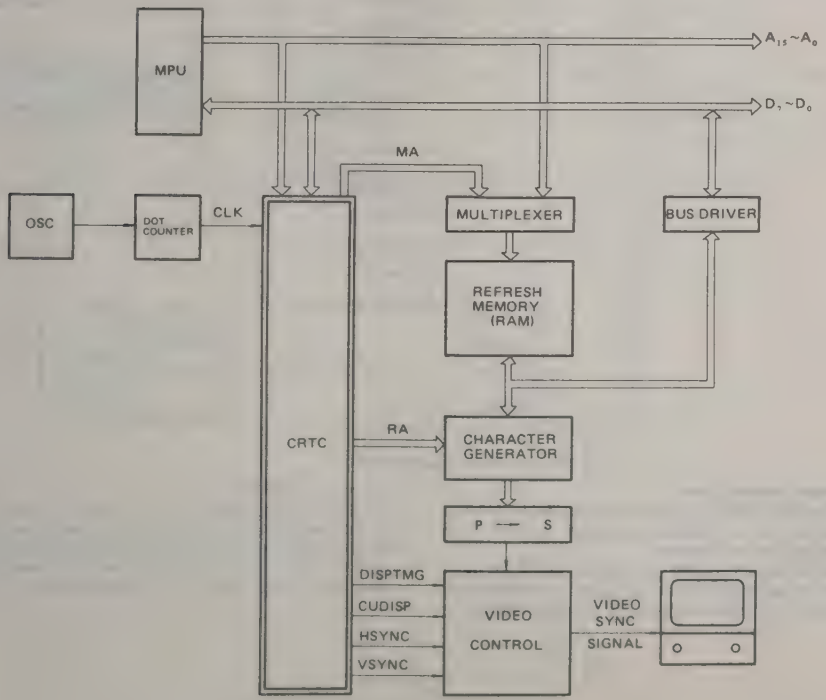


Figure 32 Monochrome Character Display

bus, while refresh memory is read out successively by the CRTC to display a static pattern on the screen. Refresh memory is accessed by both MPU and the CRTC, so it needs to change its address selectively by multiplexer. The CRTC has 14 MA (Memory Address output), but in fact some of them that are needed are used according to capacity of refresh memory.

Code output of refresh memory is provided to character generator. Character generator generates a dot pattern of a specified raster of a specified character in parallel according to code output from refresh memory and RA (Raster Address output) from the CRTC. Parallel-serial converter is normally composed of shift register to convert output of character generator into a serial dot pattern. Moreover, DISPTMG, CUDISP, HSYNC, and VSYNC are provided to video control circuit. It controls blanking for output of parallel-serial converter, mixes these signals with cursor video signal, and generates sync signals for an interface to monitor.

- **Color Character Display**

Fig. 33 shows a system of color character display. In this example, a 3-bit color control bit (R, G, B) is added to refresh memory in parallel with character code and provided to video control circuit. Video control circuit controls coloring as well as blanking and provides three primary color video signals (R, G, B signals) to CRT display device to display characters in seven kinds of color on the screen.

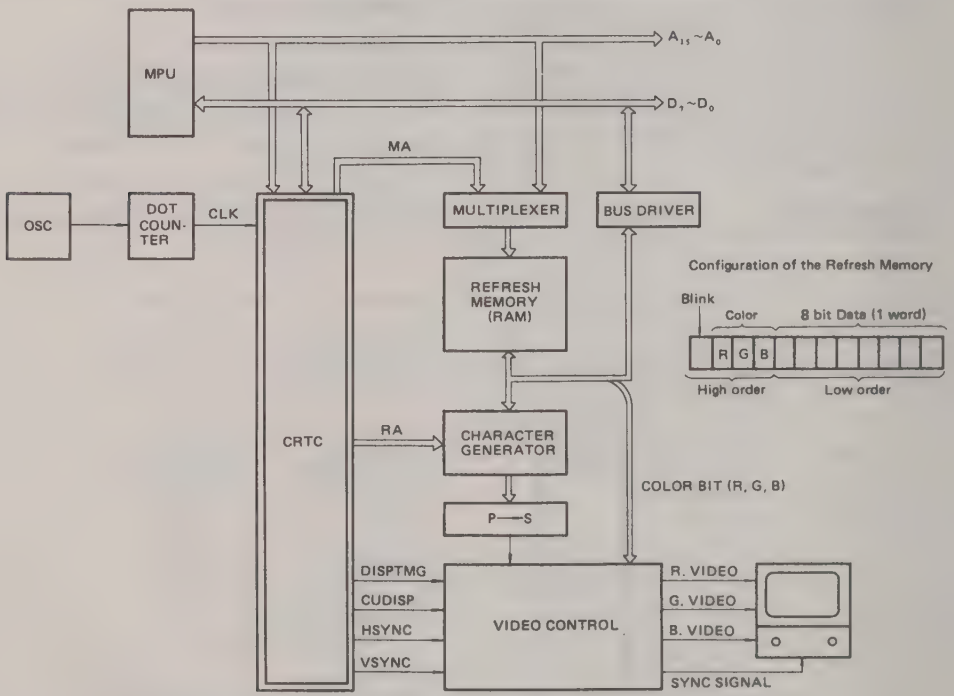


Figure 33 Color Character Display

- **Color limited Graphic Display**

Limited graphic display is to display simple figures as well as character display by combination of picture element which are defined in unit of one character.

As shown in Fig. 34, graphic pattern generator is set up in parallel with character generator and output of these generators are wire-ORed. Which generator is accessed depends on coded output of refresh memory.

In this example, graphic pattern generator adopts ROM, so only the combination of picture elements which are programmed to it is used for this graphic display system. Adopting RAM instead of ROM enables dynamically writable symbols in any combination on one display by changing the contents of them.

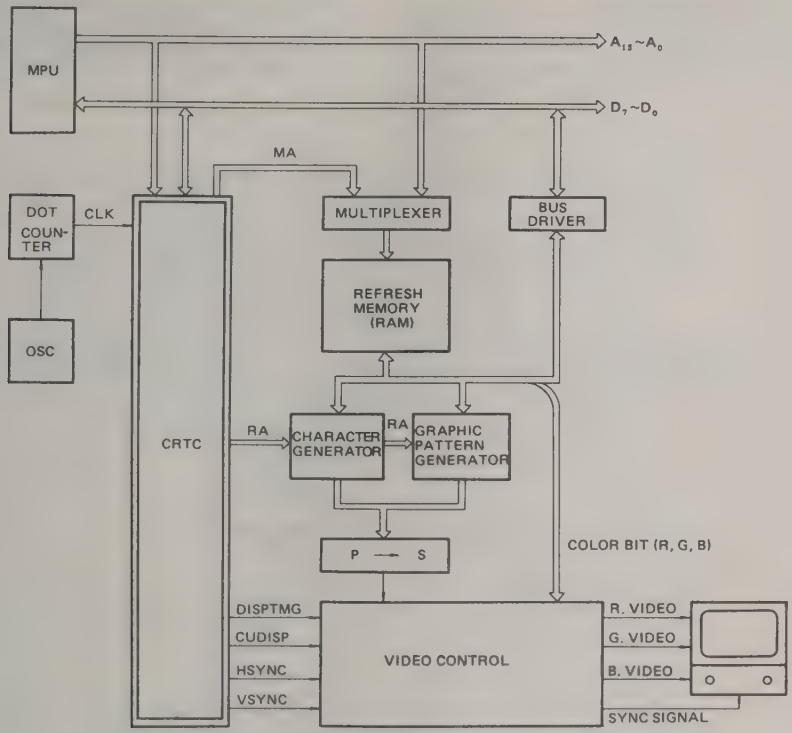


Figure 34 Color Limited Graphic Display

- **Monochrome Full Graphic Display**

Fig. 35 shows a system of monochrome full graphic display. While simple graphic display is figure display by combination of picture elements in unit of 1 picture elements, full graphic display is display of any figures in unit of 1 dot. In this case, refresh memory is dot memory that stores all the dot patterns, so its output is directly provided to parallel-serial converter to be displayed. Dot memory address to refresh the screen is set up by combination of MA and RA of CRTC.

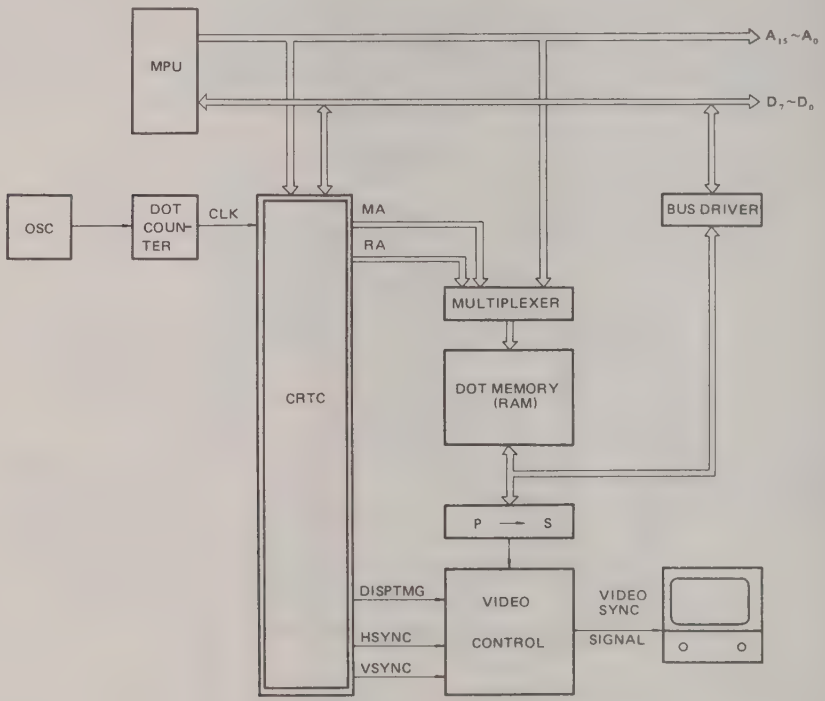


Figure 35 Monochrome Full Graphic Display

Fig. 36 shows an example of access to refresh memory by combination of MA and RA. Fig. 36 shows a refresh memory address method for full graphic display. Correspondence between dot on the CRT screen and refresh memory address is shown in Fig. 37.

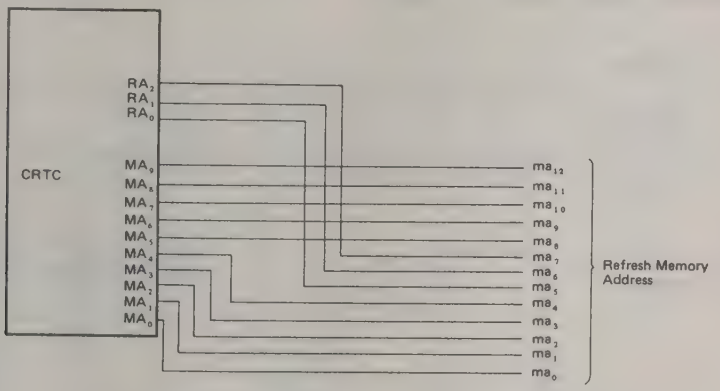


Figure 36 Refresh Memory Address Method for Full Graphic Display

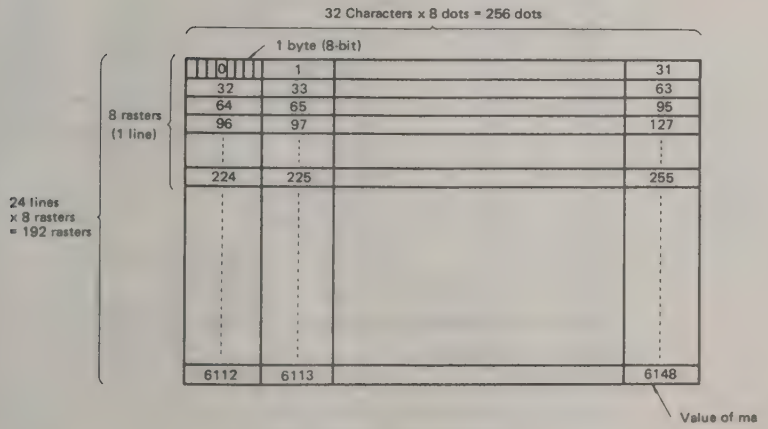


Figure 37 Memory Address and Dot Display Position on the Screen for Full Graphic Display

- **Color Full Graphic Display**

Fig. 38 shows a system of color full graphic display by 7-color display. Refresh memory is composed of three dot memories which are respectively used for red, green, and blue. These dot memories are read out in parallel at one time and their output is provided to three parallel-serial converters. Then video control circuit adds the blanking control to output of these converters and provides it to CRT display device as red, green, and blue video signals with sync signals.

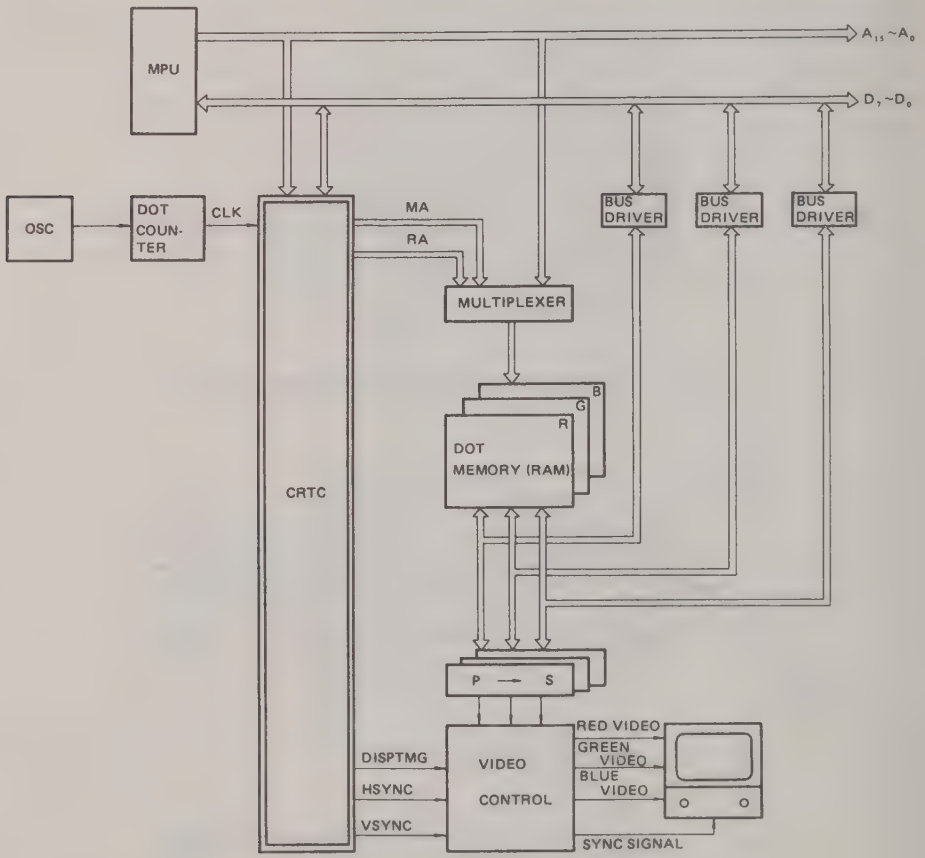


Figure 38 Color Full Graphic Display

- **Cluster Control of CRT Display**

The CRTC enables cluster control that is to control CRT display of plural devices by one CRTC. Fig. 39 shows a system of cluster control. Each display control unit has refresh memory, character generator, parallel-serial converter, and video control circuit separately, but these are controlled together by the CRTC.

In this system, it is possible for plural CRT display devices to have their own display separately.

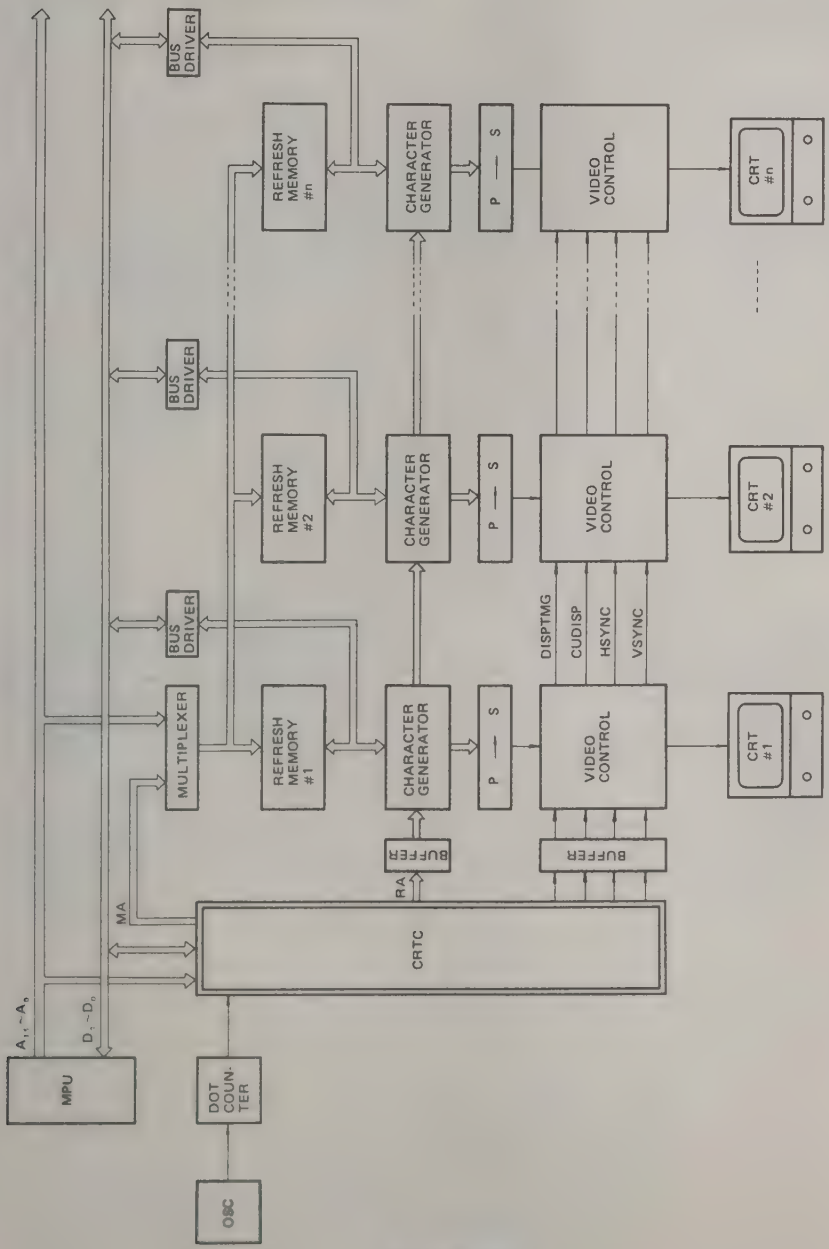


Figure 39 Cluster Control by the CRTC

■ EXAMPLES OF APPLIED CIRCUIT OF THE CRTC

Fig. 41 shows an example of application of the CRTC to monochrome character display. Its specification is shown in Table 12. Moreover, specification of CRT display unit is shown in Table 13 and initializing values for the CRTC are shown in Table 14.

Table 12 Specification of Applied Circuit

Item	Specification															
Character Format	5 × 7 Dot															
Character Space	Horizontal : 3 Dot Vertical : 5 Dot															
One Character Time	1 μs															
Number of Displayed Characters	40 characters × 16 lines = 640 characters															
Access Method to Refresh Memory	Snychronous Method (DISPTMG Read)															
Refresh Memory	640 B															
Address Map		2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
	Refresh Memory	0	0	0	0	0	0	*	*	*	*	*	*	*	*	*	*
	CRTC Address Register	0	0	0	1	0	0	×	×	×	×	×	×	×	×	×	0
	CRTC Control Register	0	0	0	1	0	0	*	×	×	×	×	×	×	×	×	1
	× · · · don't care, * · · · 0 or 1																
Synchronization Method	HVSYNC Method															

Table 13 Specification of Character Display

Item	Specification
Scan Mode	Non-interlace
Horizontal Deflection Frequency	15.625 kHz
Vertical Deflection Frequency	60.1 Hz
Dot Frequency	8 MHz
Character Dot (Horizontal × Vertical)	8 × 12 (Character Font 5 × 7)
Number of Displayed Characters (Row × Line)	40 × 16
HSYNC Width	4 μs
VSYNC Width	3 H
Cursor Display	Raster 9 ~ 10, Blink 16 Field Period
Paging, Scrolling	Not used

Table 14 Initializing Values for Character Display

Register	Name	Symbol	Initializing Value Hex (Decimal)
R0	Horizontal Total	Nht	3F (63)
R1	Horizontal Displayed	Nhd	28 (40)
R2	Horizontal Sync Position	Nhsp	34 (52)
R3	Sync Width	Nvsw, Nhsw	34
R4	Vertical Total	Nvt	14 (20)
R5	Vertical Total Adjust	Nadj	08 (8)
R6	Vertical Displayed	Nvd	10 (16)
R7	Vertical Sync Position	Nvsp	13 (19)
R8	Interlace & Skew		00
R9	Maximum Raster Address	Nr	0B (11)
R10	Cursor Start Raster	B, P, NCSTART	49
R11	Cursor End Raster	NCEND	0A (10)
R12	Start Address (H)		00 (0)
R13	Start Address (L)		00 (0)
R14	Cursor (H)		00 (0)
R15	Cursor (L)		00 (0)

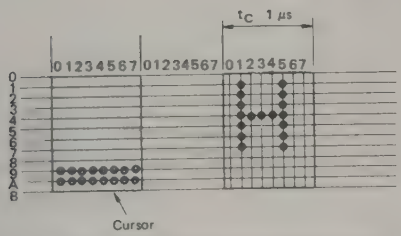
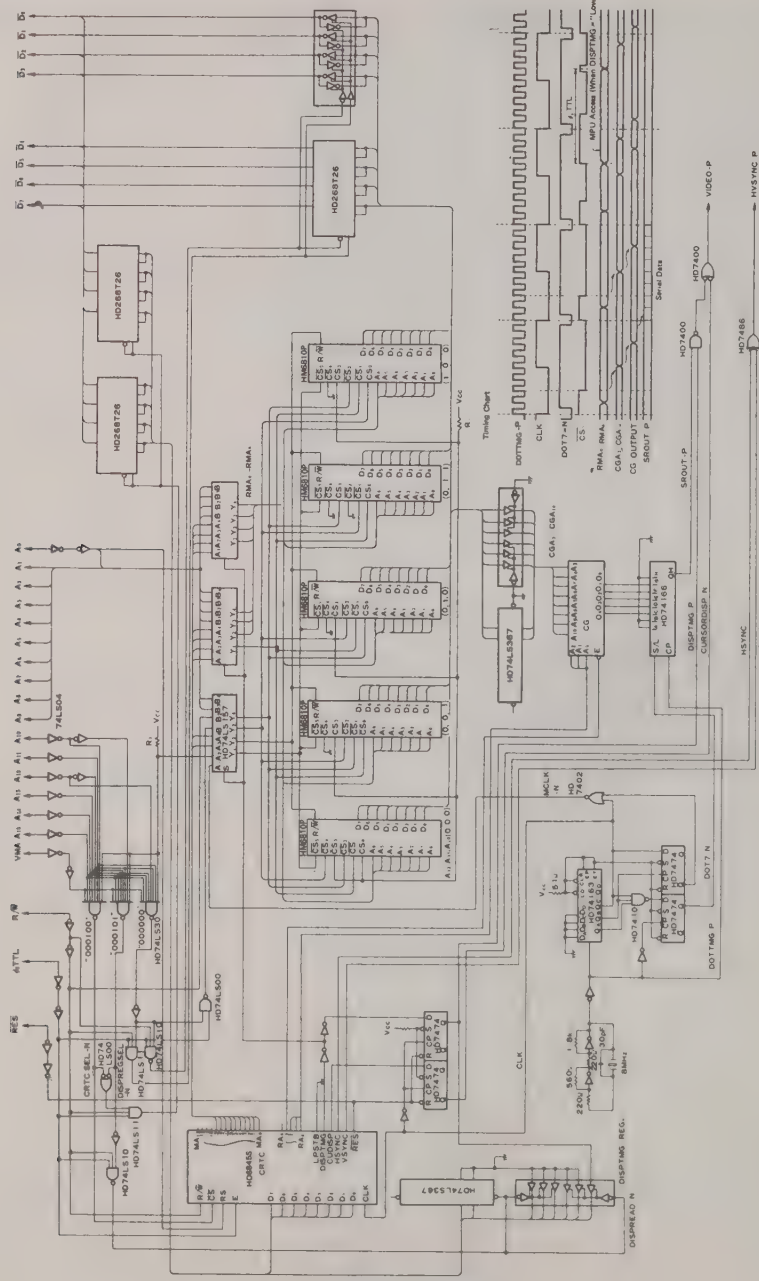


Figure 40 Non-interlace Display (Example)


HD6845
HD68A45
HD68B45
CRTC
HD74LS367
HD74LS367
HD74LS138
HD74LS00
HD74LS10
HD74LS11
HD74LS30
HD74LS00
HD74LS04
HD74LS157
HD74LS367
HD74LS374
HD74LS166
HD74LS387
HD74LS00
HD74LS02
HD74LS74
HD74LS163
HD74LS367
HD74LS10
HD74LS00
HD74LS86
HD74LS00
HD74LS00
HD74LS00

■ **DISPLAY SEQUENCE AFTER $\overline{RES}$ RELEASE OF HD6845S**

HD6845S starts the display operation immediately after the release of $\overline{RES}$. The operation at the first field is different from the normal subsequent display operation.

[Operation at the first field after the $\overline{RES}$ release]

(1) DISPTMG and CUDISP are not output. (They remain at "Low" level. The display is inhibited.)

(2) The data programmed in the start address register is not used. (MA and RA start at "0".)

(3) The sequences are shown in the following figures.

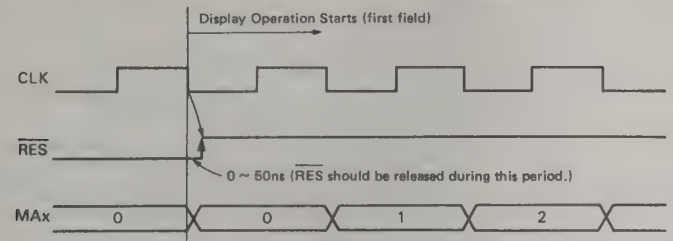


Figure 42 $\overline{RES}$ Release Sequence

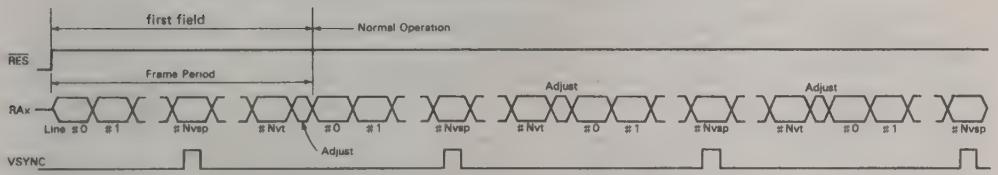
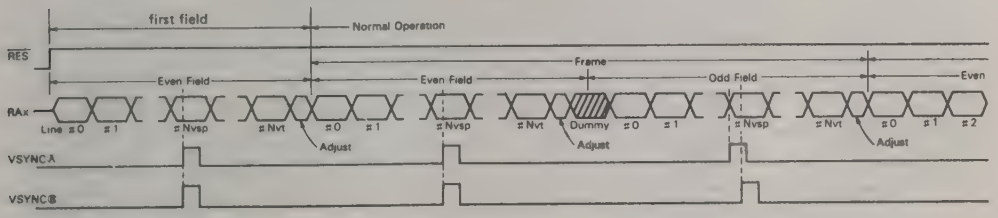
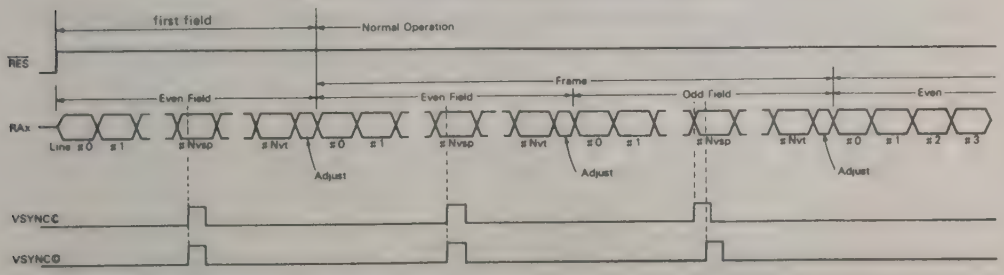


Figure 43 $\overline{RES}$ Release Sequence in The Non-interlace Mode



VSYNCⒶ : Interlace Sync Control
Interlace Sync & Video Control (Nr+2=Even)
Interlace Sync & Video Control (Nr+2=Odd, Nvt=Odd, Nvsp=Even)
VSYNCⒷ : Interlace Sync & Video Control (Nr+2=Odd, Nvt=Odd, Nvsp=Odd)

Figure 44 $\overline{RES}$ Release Sequence in The Interlace Mode (1)



VSYNCⒸ : Interlace Sync & Video Control (Nr+2=Odd, Nvt=Even, Nvsp=Even)
VSYNCⒹ : Interlace Sync & Video Control (Nr+2=Odd, Nvt=Even, Nvsp=Odd)

Figure 45 $\overline{RES}$ Release Sequence in The Interlace Mode (2)

■ ANOMALOUS OPERATIONS IN HD6845S CAUSED BY REWRITING REGISTERS DURING THE DISPLAY OPERATION*

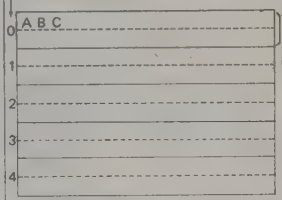
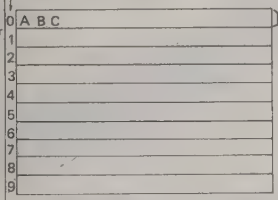
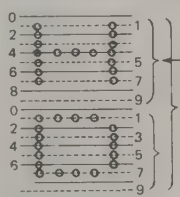
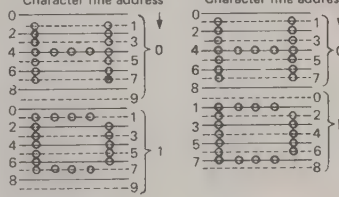
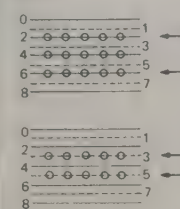
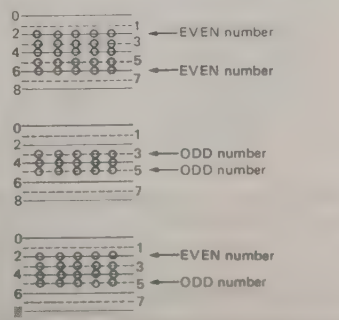
Register #	Register Name	Anomalous operations caused by rewriting registers & Conditions to avoid those operations	Rewriting** OK or NG
R0	Horizontal Total	The horizontal scan period is disturbed.	X
R1	Horizontal Displayed	There are some cases where the width of DISPTMG becomes shorter than the programmed value at the moment of a rewrite operation. An error operation occurs only during one raster period.	○
R2	Horizontal Sync Position	There are some cases where HSYNC is placed on the position different from the programmed value or the noise is output.	X
R3	Sync Width	When a rewrite operation is performed at a "High" level on HSYNC pulse or VSYNC pulse, there are some cases where the width pulse becomes shorter than the programmed value at the moment of a rewrite operation.	Δ
R4	Vertical Total	When a rewrite operation is performed during the last raster period in the line, there is a possibility that the disturbance occurs during the vertical scan period. There is no problem of a rewrite operation during raster period except this period.	Δ
R5	Vertical Total Adjust	When a rewrite operation is performed in the last character time of the raster period, there are some cases where the numbers of Adjust Raster, specified by program, are not added. (Only during the adjust raster period)	Δ
R6	Vertical Displayed	After the moment of a rewrite operation, there are some cases where the Display is inhibited. However, the display according to the programmed value is performed from the next field.	○
R7	Vertical Sync Position	There are some cases where VSYNC is placed on the position different from the programmed value or the noise is output.	X
R8	Interlace & Skew	Neither scan mode bit nor skew bit is rewritten dynamically. Dynamic Rewrite into scan mode bit and skew bit is prohibited.	X
R9	Maximum Raster Address	The internal operation will be disordered by a rewrite operation.	X
R10	Cursor Start Raster	When a rewrite operation is performed in the last character time of the raster period, there are some cases where the jitter occurs on the cursor raster or the cursor is not displayed correctly. There is also a possibility that the blink rate becomes temporally shorter than usual.	Δ
R11	Cursor End Raster	When a rewrite operation is performed in the last character time of the raster period, there are some cases where the jitter occurs on the cursor raster or the cursor is not displayed correctly. Moreover, there are also some cases where the blink rate becomes temporally shorter than normal operation.	Δ
R12	Start Address (H)	R12 and R13 are used in the last raster period of the field. A rewrite operation can be performed except during this period. However, when R12 and R13 are rewritten in each field separately, the display operation, whose start address is determined temporally by programming sequence, will be performed. A rewrite operation should be performed during the horizontal/vertical display period.	○
R13	Start Address (L)		○
R14	Cursor (H)	When a rewrite operation is performed during the display period, there are some cases where the cursor is temporally displayed at the address different from the programmed value. A rewrite operation should be performed during the horizontal/vertical retrace period. Also, when R14 and R15 are rewritten in each field separately, the cursor is displayed temporally at the temporal address determined by programming sequence.	○
R15	Cursor (L)		○

* : . . . . means temporary abnormal operations in rewriting the internal register during the display operation. Normally, after a rewrite operation, the LSI performs the specified display operation from the next field.
(The operations in this table are outside our guarantee and are regarded as materials for reference.)

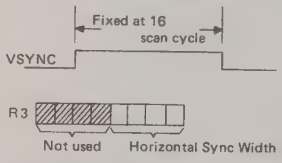
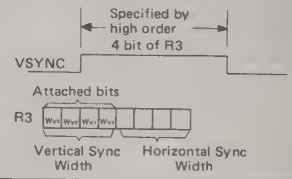
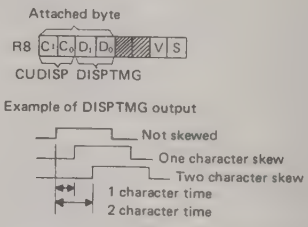
** ○ . . . . A rewrite operation is possible without affecting the screen in the display so much.
Δ . . . . If conditions are satisfied, a rewrite operation is possible. If conditions are not satisfied, there are some cases where a flicker and so on occur temporally.
X . . . . When a rewrite operation is performed, there are some cases where a flicker and so on occur temporally.

■ COMPARISON BETWEEN HD6845S AND HD6845

• Comparison of function between HD6845S and HD6845

No.	Functional Difference		HD6845	HD6845S
1	Interlace Sync & Video Mode Display	Programming Method of Number of Vertical Characters	Character line address  In HD6845, number of characters is vertically programmed in unit of two lines, as illustrated above. (Number of vertical total characters, Number of vertical displayed characters, Vertical Sync Position) Example of above figure Programmed number into Vertical Displayed Register = 5	Character line address  In HD6845S, number of characters is vertically programmed in unit of one line, as illustrated above. (Number of vertical total characters, Number of vertical displayed characters, Vertical Sync Position) Example of above figure Programmed number into Vertical Displayed Register = 10
		Number of Rasters Per Character Line	Only even number can be specified.  Number of raster = 10 scan line (specified) However, number which is programmed into register is calculated at follows. Programmed number (Nr) = (Number specified) – 1	Both even number and odd number can be specified. Character line address  When number of raster per character line is EVEN. Number of raster = 10 scan line (specified) When number of raster per character line is ODD. Number of raster = 9 scan line (specified) However, number which is programmed into register is calculated as follows. Programmed number (Nr) = (Number specified) – 2
		ırsor Display	Cursor is displayed in either EVEN field or ODD field. 	Cursor is displayed in both EVEN field and ODD field. 

(to be continued)

No.	Functional Difference	HD6845	HD6845S
2	Vertical Sync Pulse Width (VSYNC output)	Fixed at 16 raster scan cycle (16H) 	Programmable (1 ~ 16 raster scan cycle) 
3	SKEW Function	Not included 	SKEW function is newly included in DISPTMG, CUDISP signals. 
4	Start Address Register	Impossible to READ	Possible to READ
5	RESET Signal ($\overline{RES}$)	$MA_0 \sim MA_{13}$ Output, $RA_0 \sim RA_4$ Output ..... Synchronous reset Other Outputs ..... Asynchronous reset Output signals of $MA_0 \sim MA_{13}$, $RA_0 \sim RA_4$, synchronizing with DLK "Low" level, go to "Low" level, after $\overline{RES}$ has gone to "Low" Other outputs go to "Low" immediately after $\overline{RES}$ has gone to "Low" level	$MA_0 \sim MA_{13}$ Output, $RA_0 \sim RA_4$ Output, Other Outputs ..... Asynchronous reset Output signals of $MA_0 \sim MA_{13}$, $RA_0 \sim RA_4$ and others go to "Low" level immediately after $\overline{RES}$ has gone to "Low" level.

- **Comparison of Timing Signal between HD6845S and HD6845**

No.	Item	Symbol	HD6845			HD6845S			Unit
			min	typ	max	min	typ	max	
1	Clock Cycle Time	t_{cycC}	330	–	–	270	–	–	ns
2	Clock "High" Pulse Width	PW_{CH}	150	–	–	130	–	–	ns
3	Clock "Low" Pulse Width	PW_{CL}	150	–	–	130	–	–	ns
4	Rise and Fall Time for Clock Input	t_{Cr}, t_{Cf}	–	–	15	–	–	20	ns
5	Horizontal Sync Delay Time	t_{HSD}	–	–	250	–	–	200	ns
6	Light Pen Strobe Pulse Width	PW_{LPH}	80	–	–	60	–	–	ns
7	Light Pen Strobe Uncertain Time of Acceptance	t_{LPD1}	–	–	80	–	–	70	ns
		t_{LPD2}	–	–	10	–	–	0	ns

■ COMPATIBILITY OF HD6845S AND HD6845

Non-interlace mode control
Interlace sync mode control
} : Fully compatible with HD6845*
HD6845 can be directly replaced by HD6845S in these modes.

Interlace sync & Video mode control :
Not compatible with HD6845 in regard to programming and data for vertical direction need to be changed.

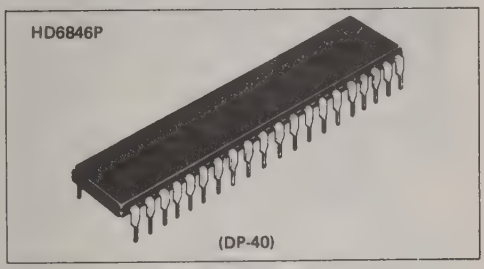
* The functions added to HD6845S utilize undefined bits of the Control Register in HD6845. If "0" is programmed to the undefined bits in the initial set, it is possible to replace HD6845 with HD6845S without changing the parameters.

Note) The restriction on programming of HD6845S and HD6845 should be taken into consideration.

HD6846
COMBO (Combination ROM I/O Timer)

The HD6846 combination chip provides the means, in conjunction with the HD6802, to develop a basic 2-chip microcomputer system. The HD6846 consists of 2048 bytes of mask-programmable ROM, an 8-bit bidirectional data port with control lines, and a 16-bit programmable timer-counter.

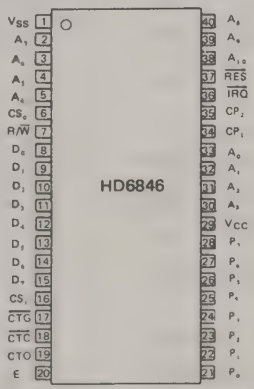
This device is capable of interfacing with the HD6802 (basic HD6800, clock and 128 bytes of RAM) as well as the HD6800 if desired. No external logic is required to interface with most peripheral devices.



■ FEATURES

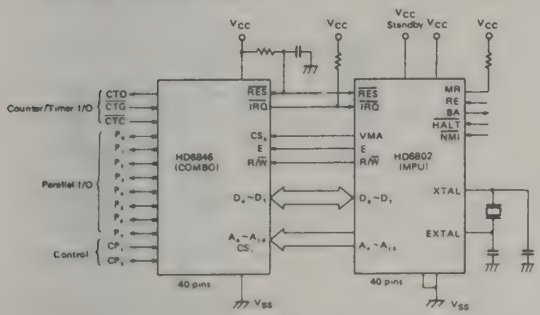
- 2048 Bytes of Mask-Programmable ROM
- 8-Bit Bidirectional Data Port for Parallel Interface plus Two Control Lines
- Programmable Interval Timer-Counter Functions
- Programmable I/O Peripheral Data, Control and Direction Registers
- Compatible with the Complete HMCS6800 Microcomputer Product Family
- TTL-Compatible Data and Peripheral Lines
- Single 5-Volt Power Supply
- Compatible with MC6846

■ PIN ARRANGEMENT



(Top View)

■ TYPICAL MICROCOMPUTER



This is a block diagram of a typical cost effective microcomputer. The MPU is the center of the microcomputer system and is shown in a minimum system interfacing with a ROM combination chip. It is not intended that this system be limited to this function but that it be expandable with other parts in the HMCS6800 Microcomputer family.

■ ABSOLUTE MAXIMUM RATINGS

Item	Symbol	Value	Unit
Supply Voltage	V_{CC}*	−0.3 ~ +7.0	V
Input Voltage	V_{in}*	−0.3 ~ +7.0	V
Operating Temperature	T_{opr}	−20 ~ +75	°C
Storage Temperature	T_{stg}	−55 ~ +150	°C

* With respect to V_{SS} (SYSTEM GND)

(NOTE) Permanent LSI damage may occur if maximum ratings are exceeded. Normal operation should be under recommended operating conditions. If these conditions are exceeded, it could affect reliability of LSI.

■ RECOMMENDED OPERATING CONDITIONS

Item	Symbol	min	typ	max	Unit
Supply Voltage	V_{CC}*	4.75	5.0	5.25	V
Input Voltage	V_{IL}*	−0.3	–	0.8	V
	V_{IH}*	2.0	–	V_{CC}	V
Operating Temperature	T_{opr}	−20	25	75	°C

* With respect to V_{SS} (SYSTEM GND)

■ ELECTRICAL CHARACTERISTICS

● DC CHARACTERISTICS (V_{CC}=5.0V±5%, V_{SS}=0V, Ta=−20~+75°C, unless otherwise noted.)

Item		Symbol	Test Condition	min	typ	max	Unit
Input "High" Voltage	All Inputs	V_{IH}		2.0	–	V_{CC}	V
Input "Low" Voltage	All Inputs	V_{IL}		−0.3	–	0.8	V
Clock Overshoot/Undershoot	Input "High" Level	V_{OS}		V_{CC} −0.5	–	V_{CC} +0.5	V
	Input "Low" Level			V_{SS} −0.5	–	V_{SS} +0.5	
Input Leakage Current	R/$\overline{W}$, $\overline{RES}$, $\overline{CS_0}$, $\overline{CS_1}$, CP_1, $\overline{CTG}$, $\overline{CTC}$, E, A_0~A_{10}	I_{in}	V_{in} = 0 ~ 5.25V	−2.5	–	2.5	μA
Three-State (Off State) Input Current	D_0~D_7, P_0~P_7, CP_2	I_{TSI}	V_{in} = 0.4 ~ 2.4V	−10	–	10	μA
Output "High" Voltage	D_0~D_7	V_{OH}	I_{OH} = −205μA	2.4	–	–	V
	CP_2, P_0~P_7		I_{OH} = −200μA	2.4	–	–	V
	CTO		I_{OH} = −200μA	2.4	–	–	V
Output "Low" Voltage	D_0~D_7	V_{OL}	I_{OL} = 1.6mA	–	–	0.4	V
	Other Outputs		I_{OL} = 3.2mA	–	–	0.4	V
Output "High" Current (Sourcing)	D_0~D_7	I_{OH}	V_{OH} = 2.4V	−205	–	–	μA
	CTO, CP_2, P_0~P_7		V_{OH} = 2.4V	−200	–	–	μA
Output "High" Current (Sourcing) (the current for driving other than TTL, e.g., Darlington Base)	CP_2, P_0~P_7	I_{OH}	V_{OH} = 1.5V	−1.0	–	−10	mA
Output "Low" Current (Sinking)	D_0~D_7	I_{OL}	V_{OL} = 0.4V	1.6	–	–	mA
	Other Outputs			3.2	–	–	
Output Leakage Current (Off State)	$\overline{IRQ}$	I_{LOH}	V_{OH} = 2.4V	–	–	10	μA
Power Dissipation		P_D		–	–	800	mW
Capacitance	E	C_{in}	V_{CC} = 0V, V_{in} = 0V, T_a = 25°C, f = 1MHz	–	–	20	pF
	D_0~D_7	C_{in}		–	–	12.5	pF
	P_0~P_7, CP_2, CTO	C_{out}		–	–	10	pF
	A_0~A_{10}, R/$\overline{W}$	C_{in}		–	–	7.5	pF
	$\overline{RES}$, $\overline{CS_0}$, CS_1, CP_1, $\overline{CTG}$	C_{in}		–	–	10	pF
	$\overline{IRQ}$	C_{out}		–	–	7.5	pF

- **AC CHARACTERISTICS** (V_{CC}=5.0V±5%, V_{SS}=0V, Ta=-20~+75°C, unless otherwise noted.)

1. BUS TIMING

Item	Symbol	Test Condition	min	typ	max	Unit
Enable Cycle Time	t_{cycE}	Fig. 1	1.0	–	10	μs
Enable Pulse Width, "Low"	PW_{EL}		430	–	4500	ns
Enable Pulse Width, "High"	PW_{EH}		430	–	4500	ns
Address Set Up Time	t_{AS}		140	–	–	ns
Data Delay Time	t_{DDR}		–	–	320	ns
Data Hold Time	t_H		10	–	–	ns
Address Hold Time	t_{AH}		10	–	–	ns
Enable Rise and Fall Time	t_{Ef},t_{Er}		–	–	25	ns
Data Set Up Time	t_{DSW}		195	–	–	ns
Reset "Low" Time	t_{RL}		2	–	–	μs
Interrupt Release Time	t_{IR}	Fig. 2	–	–	1.6	μs

2. PARALLEL PERIPHERAL I/O LINE TIMING

Item	Symbol	Test Condition	min	typ	max	Unit
Peripheral Data Setup Time	t_{PDSU}	Fig. 3	200	–	–	ns
Rise and Fall Times CP_1, CP_2	t_{Pr}, t_{pf}	Fig. 5	–	–	1.0	μs
Delay Time E to CP_2 Fall	t_{CP2}	Fig. 4	–	–	1.0,	μs
Delay Time I/O Data CP_2 Fall	t_{DC}		20	–	–	ns
Delay Time E to CP_2 Rise	t_{RS1}		–	–	1.0	μs
Delay Time CP_1 to CP_2 Rise	t_{RS2}	Fig. 5	–	–	2.0	μs
Peripheral Data Delay	t_{PDW}	Fig. 4	–	–	1.0	μs
Peripheral Data Setup Time for Latch	t_{PSU}	Fig. 9	100	–	–	ns
Peripheral Data Hold Time for Latch	t_{PDH}		15	–	–	ns

3. TIMER/COUNTER LINE TIMING

Item	Symbol	Test Condition	min	typ	max	Unit
$\overline{CTC}$, $\overline{CTG}$ Rise and Fall Time	t_{Cr}, t_{Cf}	Fig. 6	–	–	100	ns
$\overline{CTC}$, $\overline{CTG}$ Pulse Width, "High" (Asynchronous Mode)	t_{PWH}		t_{cycE} +250	–	–	ns
$\overline{CTC}$, $\overline{CTG}$ Pulse Width, "Low" (Asychronous Mode)	t_{PWL}		t_{cycE} +250	–	–	ns
$\overline{CTC}$, $\overline{CTG}$ Setup Time (Synchronous Mode)	t_{su}	Fig. 7	200	–	–	ns
$\overline{CTC}$, $\overline{CTG}$ Hold Time (Synchronous Mode)	t_{hd}		50	–	–	ns
CTO Delay Time	t_{CTO}	Fig. 8	–	–	1.0	μs

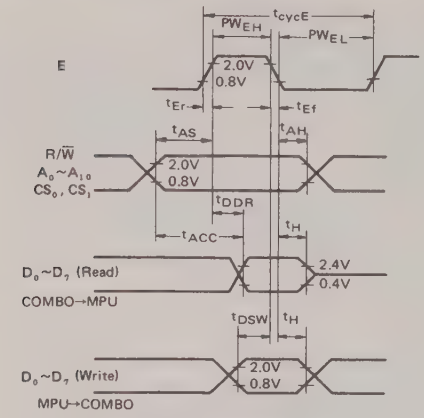


Figure 1 Bus Read/Write Timing

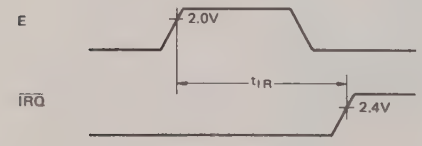


Figure 2 $\overline{IRQ}$ Release Time

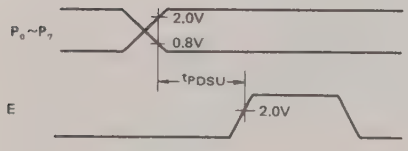


Figure 3 Peripheral Data Set Up Time

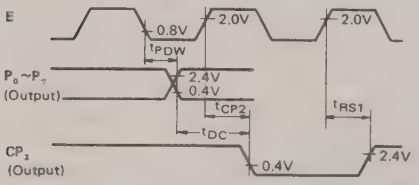


Figure 4 Peripheral Data and CP_2 (Output) Delay Time

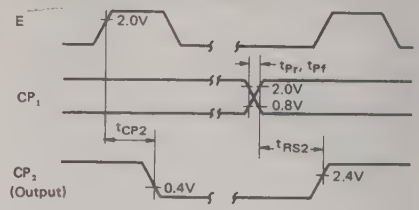


Figure 5 CP_2 (Output) Delay Time

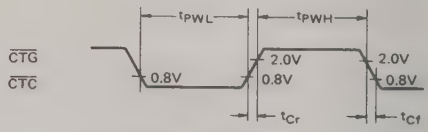


Figure 6 $\overline{CTG}$, $\overline{CTC}$ Pulse Width

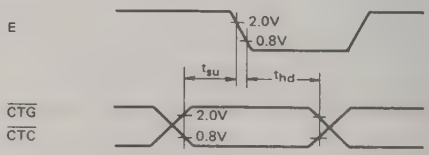


Figure 7 $\overline{CTG}$, $\overline{CTC}$ Setup Time and Hold Time

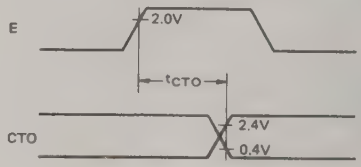


Figure 8 CTO Delay Time

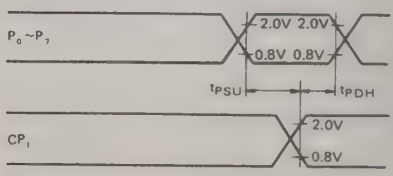


Figure 9 Peripheral Port Latch Setup and Hold Time

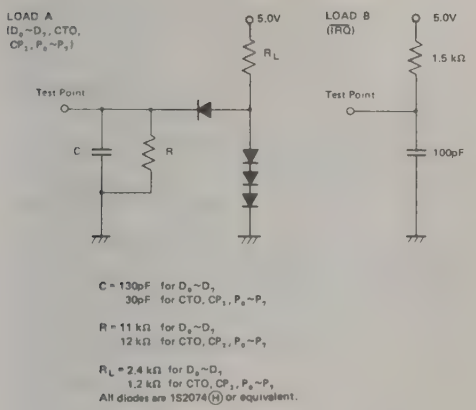


Figure 10 Bus Timing Test Loads

■ GENERAL DESCRIPTION

The HD6846 combination chip may be partitioned into three functional operating sections: programmed storage, timer-counter functions, and a parallel I/O port.

• Programmed Storage

The mask-programmable ROM section is similar to other ROM products of the HMCS6800 family. The ROM is organized in a 2048 by 8-bit array to provide read only storage for a minimum microcomputer system. Two mask-programmable chip selects are available for user definition.

Address inputs $A_0 \sim A_{10}$ allow any of the 2048 bytes of ROM to be uniquely addressed. Bidirectional data lines ($D_0 \sim D_7$) allow the transfer of data between the MPU and the HD 6846.

• Timer-Counter Functions

Under software control this 16-bit binary counter may be programmed to count events, measure frequencies, time intervals, or similar tasks. Internal registers associated with the I/O functions may be selected with A_0, A_1 and A_2. It may also be used for square wave generation, single pulses of controlled duration, and gated signals. Interrupts may be generated from a number of conditions selectable by software programming.

The timer/counter control register allows control of the interrupt enable, output enable, selection of an internal or external clock source, a ÷ 8 prescaler, and operating mode. Input pin $\overline{CTC}$ (counter-timer clock) will accept an asynchronous clock pulse to decrement the internal register for the counter-timer. If the divide-by-8 prescaler is used, the maximum clock rate can be four times the master clock frequency with an absolute maximum of 4 MHz. Gate input ($\overline{CTG}$) accepts an asynchronous TTL-compatible signal which may be used as a trigger or gating function to the counter-timer. A counter-timer output (CTO) is also available and is under software control being dependent on the timer control register, the gate input, and the clock source.

• Parallel I/O Port

The parallel bidirectional I/O port has functional operational characteristics similar to the B port on the HD6821 PIA. This includes 8 bidirectional data lines and two handshake control signals. The control and operation of these lines are completely software programmable.

The interrupt input (CP_1) will set the interrupt flag CSR1 of the composite status register. The peripheral control (CP_2) may be programmed to act as an interrupt input (set CSR2) or as a peripheral control output.

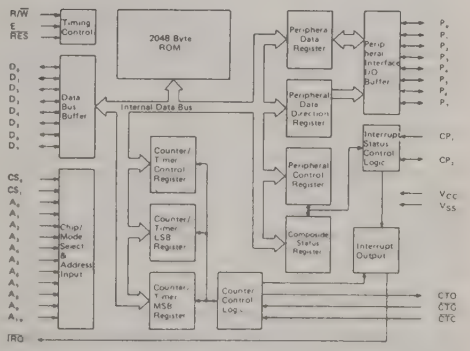


Figure 11 Combination ROM I/O Timer (COMBO)

Basic Block Diagram

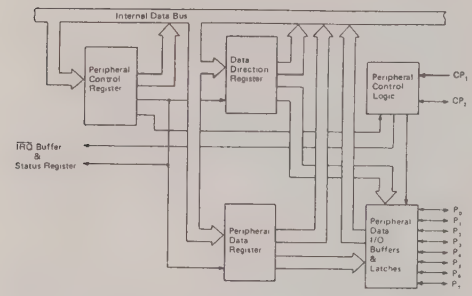


Figure 12 Parallel I/O Port Block Diagram

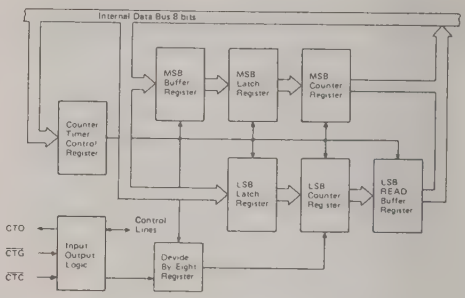


Figure 13 Timer/Counter Block Diagram

■ SIGNAL DESCRIPTION

• Bus Interface

The HD6846 interfaces to the HMCS6800 Bus via an eight bit bidirectional data bus, two Chip Select lines, a Read/Write line, and eleven address lines. These signals, in conjunction with the HMCS6800 VMA output, permit the MPU to control the HD6846.

• Bidirectional Data Bus ($D_0 \sim D_7$)

The bidirectional data lines ($D_0 \sim D_7$) allow the transfer of data between the MPU and the HD6846. The data bus output drivers are three-state devices which remain in the high-impedance (Off) state except when the MPU performs an HD6846 register or ROM read ($R/\overline{W}$ = 1 and I/O Registers or ROM selected).

• Chip Select (CS_0, CS_1)

The CS_0 and CS_1 inputs are used to select the ROM or I/O timer of the HD6846. They are mask programmed to be active "High" or active "Low" as chosen by the user.

• Address Inputs ($A_0 \sim A_{10}$)

The Address Inputs allow any of the 2048 bytes of ROM to be uniquely selected when the circuit is operating in the ROM mode. In the I/O-Timer mode, address inputs A_0, A_1, and A_2 select the proper I/O Register, while A_3 through A_{10} (together with CS_0 and CS_1) can be used as additional qualifiers in the I/O Select circuitry. (See the section on I/O-Timer Select for additional details.)

• Reset ($\overline{RES}$)

The active "Low" state of the $\overline{RES}$ input is used to initialize all register bits in the I/O section of the device to their proper values. (See the section on Initialization for Reset conditions for timer and peripheral registers.)

• Enable (E)

This signal synchronizes data transfer between the MPU and the HD6846. It also performs an equivalent synchronization function on the external clock, reset, and gate inputs of the HD6846 Timer section.

• Read/Write ($R/\overline{W}$)

This signal is generated by the MPU and is used to control the direction of data transfer on the bidirectional data pins. A "Low" level on the $R/\overline{W}$ input enables the HD6846 input buffers and data is transferred to the circuit during the E pulse when the part has been selected. A "High" level on the $R/\overline{W}$ input enables the output buffers and data is transferred to the MPU during E when the part is selected.

• Interrupt Request ($\overline{IRQ}$)

The active "Low" $\overline{IRQ}$ output acts to interrupt the MPU through logic included on the HD6846. This output utilizes an open drain configuration and permits other interrupt request outputs from other circuits to be connected in a wire-OR configuration.

• Peripheral Data ($P_0 \sim P_7$)

The peripheral data lines can be individually programmed as either inputs or outputs via the Data Direction Register. When programmed as outputs, these lines will drive two standard TTL

loads (3.2 mA). They are also capable of sourcing up to 1.0 mA at 1.5 Volts (Logic "1" output.)

When programmed as inputs, the output drivers associated with these lines enter a three-state (high impedance) mode. Since there is no internal pull-up for these lines, they represent a maximum 10μA load to the circuitry driving them – regardless of logic state.

A logic zero at the $\overline{RES}$ input forces the peripheral data lines to the input configuration by clearing the Data Direction Register. This allows the system designer to preclude the possibility of having a peripheral data output connected to an external driver output during power-up sequence.

- **Interrupt Input (CP_1)**

Peripheral input line CP_1 is an input-only that sets the Interrupt Flags of the Composite Status register. The active transition for this signal is programmed by the peripheral control register for the parallel port. CP_1 may also act as a strobe for the peripheral data register when it is used as an input latch. Details for programming CP_1 are in the section on the parallel peripheral port.

(Note)

Unexpected noise may occur on the peripheral data line when the peripheral data register is loaded with "1". This erroneous noise may occur only when peripheral data line is specified as output and the peripheral data register has already been loaded with "1". Note that peripheral data line doesn't keep "High" level continuously in the case write peripheral data register operation is executed.

- **Peripheral Control (CP_2)**

Peripheral Control line CP_2 may be programmed to act as an Interrupt input or Peripheral Control output. As an input, this line has high impedance and is compatible with standard TTL voltage levels. As an output, it is also TTL compatible and may be used as a source of 1 mA at 1.5 V to directly drive the base of a Darlington transistor switch. This line is programmed by the Peripheral Control Register.

- **Counter Timer Output (CTO)**

The Counter Timer Output is software programmable by selected bits in the timer/counter control register. The mode of operation is dependent on the Timer control register, the gate input, and the clock source. The output is TTL compatible.

- **External Clock Input ($\overline{CTC}$)**

Input pin $\overline{CTC}$ will accept asynchronous TTL voltage level signals to be used as a clock to decrement the Timer. The "High" and "Low" levels of the external clock must be stable for at least one system clock period plus the sum of the setup and hold times for the inputs. The asynchronous clock rate can vary from dc to the limit imposed by System E, setup, and hold times.

The external clock input is clocked in by Enable pulses. Three Enable periods are used to synchronize and process the external clock. The fourth Enable pulse decrements the internal counter. This does not affect the input frequency; it merely creates a delay between a clock input transition and internal recognition of that transition by the HD6846. All references to $\overline{CTC}$ inputs in this document relate to internal recognition of the input transition. Note that a clock transition which does not meet setup and hold time specifications may require an additional Enable pulse for recognition.

When observing recurring events, a lack of synchronization will result in either "System jitter" or "Input jitter" being observed on the output of the HD6846 when using an asynchronous clock and gate input signal. "System jitter" is the result of the input signals being out of synchronization with Enable, permitting signals with marginal set-up and hold time to be recognized by either the bit time nearest the input transition or subsequent bit time. "Input jitter" can be as great at the time between the negative going transitions of the input signal plus the system jitter if the first transition is recognized during one system cycle, and not recognized the next cycle or vice-versa.

- **Gate Inputs ($\overline{CTG}$)**

The input pin $\overline{CTG}$ accepts an asynchronous TTL-compatible signal which is used as a trigger or a clock gating function to the Timer. The gating input is clocked into the HD6846 by the Enable signal in the same manner as the previously discussed clock inputs. That is, a $\overline{CTG}$ transition is recognized on the fourth Enable pulse (provided setup and hold time requirements are met), and the "High" or "Low" levels of the $\overline{CTG}$ input must be stable for at least one system clock period plus the sum of setup and hold times. All references to $\overline{CTG}$ transition in this document relate to internal recognition of the input transition.

The $\overline{CTG}$ input of the timer directly affects the internal 16-bit counter. The operation of $\overline{CTG}$ is therefore independent of the ÷ 8 prescaler selection.

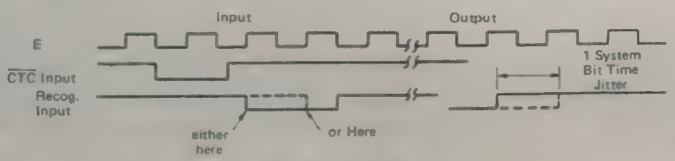
■ FUNCTIONAL SELECT CIRCUITRY

- **I/O-Timer Select Circuitry**

CS_0 and CS_1 are user programmable. Any of the four binary combinations of CS_0 and CS_1 can be used to select the ROM. Likewise, any other combination can be used to select the I/O-Timer. In addition, several address lines are used as qualifiers for the I/O-Timer. Specifically, $A_3 = A_4 = A_5$ = logical "0". A_6 can be programmed to a "1", "0", or don't care. $A_7 = A_8 = A_9 = A_{10}$ = don't care or one line only may be programmed to a logical "1". Figure 14 outlines in diagrammatic form the available chip select options.

- **Internal Addressing**

Seven I/O Register locations within the HD6846 are accessible to the MPU data bus. Selection of these registers is controlled by A_0, A_1, and A_2 (as shown in Table 1) provided the I/O timer is selected. The combination status register is Read-only; all other Registers are Read and Write.



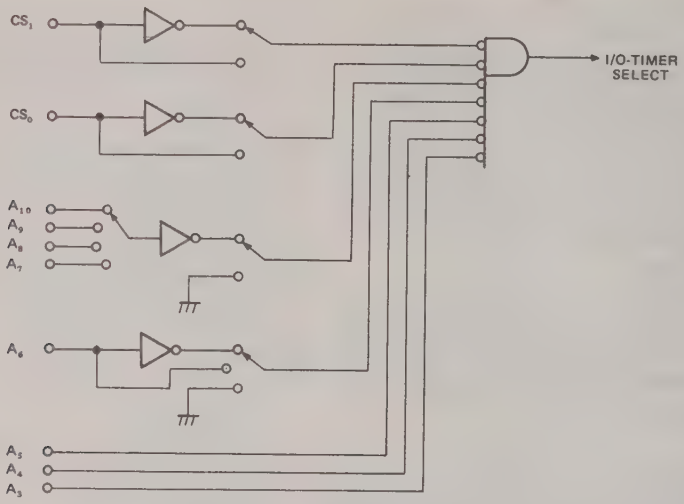


Figure 14 I/O-Timer Select Circuitry

Table 1 Internal Register Addresses

REGISTER SELECTED	A_2	A_1	A_0
Combination Status Register	0	0	0
Peripheral Control Register	0	0	1
Data Direction Register	0	1	0
Peripheral Data Register	0	1	1
Combination Status Register	1	0	0
Timer Control Register	1	0	1
Timer MSB Register	1	1	0
Timer LSB Register	1	1	1
ROM Address	×	×	×

Initialization

When the $\overline{RES}$ input has accepted a "Low" signal, all registers are initialized to the reset state. The data direction and peripheral data registers are cleared. The Peripheral Control Register is cleared except for bit 7 (the $\overline{RES}$ bit). This forces the parallel port to the input mode with Interrupts disabled. To remove the $\overline{RES}$ condition from the parallel port, a "0" must be written into the Peripheral Control Register bit 7 (PCR7).

The counter latches are preset to their maximal count, the Timer control register bits are reset to zero except for Bit 0 (TCR0 is set), the counter output is cleared, and the counter clock disabled. This state forces the timer counter to remain in an inactive state. The combination status register is cleared of all interrupt flags. During timer initialization, the reset bit (CCR0) must be cleared.

ROM

The Mask Programmable ROM section is similar in operation to other ROM products of the HMCS6800 Microprocessor family. The ROM is organized as 2048 words of 8-bits to provide read-only storage for a minimum microcomputer system. The ROM is active when selected by the unique combination of the chip select inputs.

ROM Select

The active levels of CS_0 and CS_1 for ROM and I/O select are a user programmable option. Either CS_0 or CS_1 may be programmed active "High" or active "Low", but different codes must be used for ROM or I/O select. CS_0 and CS_1 are mask programmed simultaneously with the ROM pattern. The ROM Select Circuitry is shown in Figure 15.

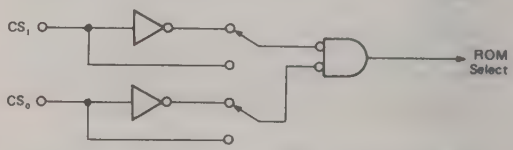


Figure 15 ROM Select Circuitry

■ TIMER OPERATION

The Timer may be programmed to operate in modes which fit a wide variety of applications. The device is fully bus compatible with the HMCS6800 system, and is accessed by Load and Store operations from the MPU.

In a typical application, the timer will be loaded by storing two bytes of data into the counter latch. This data is then transferred into the counter during a Counter Initialization cycle. The counter decrements on each subsequent clock cycle (which may be Enable or an external clock) until one of several predetermined conditions causes it to halt or recycle. Thus the timer is programmable, cyclic in nature, controllable by external inputs or MPU program, and accessible to the MPU at any time.

- **Counter Latch Initialization**

The Timer consists of a 16-bit addressable counter and two 8-bit addressable latches. The function of the latches is to store a binary equivalent of the desired count value minus one. Counter initialization results in the transfer of the latch contents of the counter. It should be noted that data transfer to the counters is always accomplished via the latches. Thus, the counter latches may be accurately described as a 16-bit "counter initialization data" storage register.

In some modes of operation, the initialization of the latches will cause simultaneous counter initialization (i.e. immediate transfer of the new latch data into the counters). It is, therefore, necessary to insure that all-16-bit of the latches are updated simultaneously. Since the HD6846 data bus is 8-bit wide, a temporary register (MSB Buffer Register) is provided for in the Most Significant Byte of the desired latch data. This is a "write-only" register selected via address lines A_0, A_1, and A_2. Data is transferred directly from the data bus to the MSB Buffer when the chip is selected, R/$\overline{W}$ is "Low", and the timer MSB register is selected (A_0 = "0", A_1 = A_2 = "1").

The lower 8-bit of the counter latch can also be referred to as a "write-only" register. Data Bus information will be transferred directly to the LSB of a counter latch when the chip is selected, R/$\overline{W}$ is "Low" and the Timer LSB Register is selected (A_0 = A_1 = A_2 = "1"). Data from the MSB Buffer will automatically be transferred into the Most Significant Byte of the counter latches simultaneously with the transfer of the Data Bus information to the Least Significant Byte of the Counter Latch. For brevity, the conditions for this operation will be referred to henceforth as a "Write Timer Latches Command."

The HD6846 has been designed to allow transfer of two bytes of data into the counter latches from any source, provided the MSB is transferred first. In many applications, the source of data will be an HMCS6800 MPU. It should therefore be noted that the 16-bit store operations of the HMCS6800 family microprocessors (STS and STX) transfer data in the order required by the HD6846. A Store Index Register instruction, for example, results in the MSB of the index register being transferred to the selected address, then the LSB of the index register being written into the next higher location. Thus, either the index register or stack pointer may be transferred directly into a selected counter latch with a single instruction.

A logic zero at the $\overline{RES}$ input also initializes the counter latches. All latches will assume maximum count (65,535) values. It is important to note that an internal reset (Bit zero of the Timer/Control Register Set) has no effect on the counter latches.

- **Counter Initialization**

Counter Initialization is defined as the transfer of data from the latches to the counter with attendant clearing of the Individual Interrupt Flag associated with the counter. Counter Initialization always occurs when a reset condition (external $\overline{RES}$ = "0" or TCR0 = "1") is recognized. It can also occur (dependent on The Timer Mode) with a Write Timer Latches command or recognition of a negative transition of the $\overline{CTG}$ input.

Counter recycling or reinitialization occurs when a clock input is recognized after the counter has reached an all-zero state. In this case, data is transferred from the Latches to the Counter, but the Interrupt Flag is unaffected.

- **Timer Control Register**

The Timer Control register (see Table 2) in the HD6846 is used to modify timer operation to suit a variety of applications. The Timer Control Register has a unique address space (A_0 = "1, A_1 = "0", A_2 = "1") and therefore may be written into at any time. The least significant bit of the Control Register is used as an Internal Reset bit. When this bit is a logic zero, all timers are allowed to operate in the modes prescribed by the remaining bits of the timer control register.

Writing "1" into Timer Control Register Bit 0 (TCR0) causes the counter to be preset with the conents of the counter latches, all counter clocks are disabled, and the timer output and interrupt flag (Status Register) are reset. The Counter Latch and Timer/Control Register are undisturbed by an Internal Reset and may be written into regardless of the state of TCR0.

Timer Control Register Bit 1 (TCR1) is used to select the clock source. When TCR1 = "0", the external clock input $\overline{CTC}$ is selected, and when TCR1 = "1", the timer uses Enable.

Table 2 Format for Timer/Counter Control Register

CONTROL REGISTER BIT	STATE	BIT DEFINITION	STATE DEFINITION
TCR0	0	Internal Reset	Timer Enabled
	1		Timer in Preset State
TCR1	0	Clock Source	Timer uses External Clock ($\overline{CTC}$)
	1		Timer uses ϕ2 System Clock
TCR2	0	÷8 Prescaler Enabler	Clock is not Prescaled
	1		Clock is prescaled by ÷8 Counter
TCR3 TCR4 TCR5	-	Operating Mode Selection	See Table 3
TCR6	0	Timer Interrupt Enable	$\overline{IRQ}$ Masked from Timer
	1		$\overline{IRQ}$ Enabled from Timer
TCR7	0	Timer Output Enable	Counter Output (CTO) Set "Low"
	1		Counter Output Enabled

Table 3 Counter/Timer Operation Modes

Mode	TCR3	TCR4	TCR5	Counter Initialization	Counter Enable	Counter Clock "CC"	Interrupt Flag	
							Set	Clear
Continuous Mode	0	0	0	$\overline{G}\downarrow$ +W+R	($\overline{G}$=Low) · $\overline{R}$	CE · C	TO	RS–RT or CI
	0	1	0	$\overline{G}\downarrow$ +R	($\overline{G}$=Low) · $\overline{R}$	CE · C	TO	RS–RT or CI
Cascaded Single Shot Mode	0	0	1	$\overline{G}\downarrow$ +W+R	$\overline{R}$	CE · C	TO	RS–RT or CI
Normal Single Shot Mode	0	1	1	$\overline{G}\downarrow$ +R	$\overline{R}$	CE · C	TO	RS–RT or CI
Frequency Comparison Mode	1	0	0	$(\overline{CE}+TOF\cdot CE)\cdot\overline{G}\downarrow\cdot\overline{I}$ +R	CE set=$\overline{G}\downarrow\cdot\overline{W}\cdot\overline{R}\cdot\overline{T}$ CE reset=W+R+I	CE · C	$\overline{G}\downarrow$ before TO	RS–RT or CI or W
	1	0	1	$\overline{G}\downarrow\cdot\overline{I}$+R	CE set=$\overline{G}\downarrow\cdot\overline{W}\cdot\overline{R}\cdot\overline{T}$ CE reset=W+R+I	CE · C	$\overline{G}\downarrow$ before TO	RS–RT or CI or W
Pulse Width Comparison Mode	1	1	0	$\overline{G}\downarrow\cdot\overline{I}$+R	CE set=$\overline{G}\downarrow\cdot\overline{W}\cdot\overline{R}\cdot\overline{T}\cdot G$ CE reset=W+R+I+($\overline{G}$=High)	CE · C	$\overline{G}\uparrow$ before TO	RS–RT or CI or W
	1	1	1	$\overline{G}\downarrow\cdot\overline{I}$+R	CE set=$\overline{G}\downarrow\cdot\overline{W}\cdot\overline{R}\cdot\overline{T}\cdot G$ CE reset=W+R+I+($\overline{G}$=High)	CE · C	$\overline{G}\uparrow$ before TO	RS–RT or CI or W

R = External $\overline{RES}$ or Internal Reset TCR0
W = Write Timer Latch
I = Interrupt Flag
$\overline{G}$ = $\overline{CTG}$
C = Clock selected in the internal register
$\overline{G}\downarrow$ = Negative transition of $\overline{CTG}$ signal
$\overline{G}\uparrow$ = Positive transistion of $\overline{CTG}$ signal
RS–RT = Read Operation of Timer Counter after the read of Status Register (Normal operation to clear the interrupt)
CI = Counter Initialization (Internal Signal)
TOF = Time Out Flag (Set by $\overline{CI}$·TO, Reset by CI)
TO = Counter Time Out

Timer Control Register Bit 2 (TCR2) enables the ÷ 8 prescaler (TCR2 = "1"). In this mode, the clock frequency is divided by eight before being applied to the counter. When TCR2 = "0" Enable is applied directly to the counter.

TCR3, 4, 5 select the Timer Operating Mode, and are discussed in the next section.

Timer Control Register Bit 6 (TCR6) is used to mask or enable the Timer Interrupt Request. When TCR6 = "0", the Interrupt Flag is masked from the timer. When TCR6 = "1", the Interrupt Flag is enabled into Bit 7 of the Composite Status Register (Composite IRQ Bit), which appears on the $\overline{IRQ}$ output pin.

Timer Control Register Bit Seven (TCR7) has a special function when the timer is in the Cascaded Single Shot mode. (This function is explained in detail in the section describing the mode.) In all other modes, TCR7 merely acts as an output enable bit. If TCR7 = "0", the Counter Timer Output (CTO) is forced "Low". Writing a logic one into TCR7 enables CTO.

- **Timer Operating Modes**

The HD6846 has been designed to operate effectively in a wide variety of applications. This is accomplished by using three bits of the control register (TCR3, TCR4, and TCR5) to define different operating modes of the Timer, outlined in Table 3.

- **Continuous Operating Mode (TCR3 = 0, TCR5 = 0)**

The timer may be programmed to operate in a continuous counting mode by writing zeros into bits 3 and 5 of the timer control register. Assuming that the timer output is enabled (TCR7 = "1"), a square wave will be generated at the Timer Output CTO.(See Table 4).

Table 4 Continuous Operating Modes

CONTINUOUS MODE
(TCR3=0, TCR7=1, TCR5=0)

CONTROL REGISTER	INITIALIZATION/OUTPUT WAVEFORMS	
TCR4	Counter	CTO
0	Initialization $\overline{G}\downarrow$ +W+R	←(N+1) (T)→ ←(N+1) (T)→ ←(N+1) (T)→
1	$\overline{G}\downarrow$ +R	V_{OH} / V_{OL}
		to TO* TO* TO*

$\overline{G}\downarrow$ = Negative Transition $\overline{CTG}$ Input.
W = Write Timer Latches Command.
R = Timer Reset (TCR0=1 or External $\overline{RES}$=0)
N = 16 Bit Number in Counter Latch.
T = Period of Clock Input to Counter.
to = Counter Initialization Cycle.
TO = Counter Time Out (All Zero Condition.)
* Point at which an interrupt may occur.

Either a Timer Reset (TCR0 = "1" or External $\overline{RES}$ = "0") condition or internal recognition of a negative transition of the $\overline{CTG}$ input results in Counter Initialization. A Write Timer Latches command can be selected as a Counter Initialization signal by clearing TCR4.

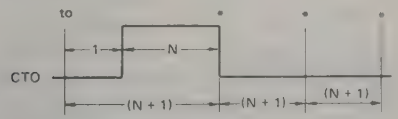
The discussion of the Continuous Mode has assumed the application requires an output signal. It should be noted the Timer operates in the same manner with the output disabled (TCR7 = "0"). A Read Timer Counter command is valid regardless of the state of TCR7.

- **Normal Single-Shot Timer Mode (TCR3 = 0, TCR4 = 1, TCR5 = 1)**

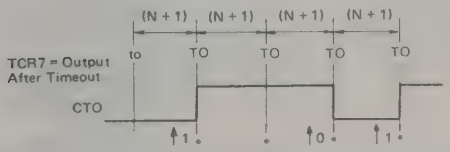
This mode is identical to the Continuous Mode with two exceptions. The first of these is obvious from the name – the output returns to a "Low" level after the initial Time Out and remains "Low" until another Counter Initialization cycle occurs. The output waveform (CTO) is shown in Figure 16.

As indicated in Figure 16, the internal counting mechanism remains cyclical in the Single-Shot Mode. Each Time Out of the counter results in the setting of an Individual Interrupt Flag and re-initialization of the counter.

The second major difference between the Single-Shot and Continuous modes is that the internal counter enable is not dependent on the $\overline{CTG}$ input level remaining in the "Low" state for the Single-Shot mode. Aside from these differences, the two modes are identical.



(A) NORMAL SINGLE-SHOT MODE OUTPUT WAVEFORM



(B) CASCADED SINGLE-SHOT MODE OUTPUT WAVEFORM

1 = Write a "1" into TCR7
0 = Write a "0" into TCR7

*Point at which an interrupt may occur.

(NOTE) All time intervals shown above assume the Gate ($\overline{CTG}$) and Clock ($\overline{CTC}$) signals are synchronized to Enable with the specified setup and hold time requirements.

Figure 16 Single-Shot Modes

- **Cascaded Single-shot Mode (TCR3=0, TCR4=0, TCR5=1)**

This mode is identical to the single-shot mode with two exceptions. First, the output waveform does not return to a "Low" level and remain "Low" after timeout. Instead, the output level remains at its initialized level until it is re-programmed and changed by timeout. The output level may be changed at any timeout or may have any number of timeouts between changes.

The second difference is the method used to change the output level. Timer Control Register Bit 7 (TCR7) has a special function in this mode. The timer output (CTO) is equal to TCR7 clocked by timeout. At every timeout, the content of TCR7 is clocked to and held at the CTO output. Thus, output pulses of length greater than one timer cycle can be generated by cascading timer cycles and counting timeouts with a software program (See Figure 16).

An interrupt is generated at each timeout. To cascade timer cycles, the MPU would need an interrupt routine to: 1) count each timeout and determine when to change TCR7: 2) write into TCR7 the state corresponding to the next desired state of the output waveform (only necessary during the last timer cycle before the output is to change state): and 3) clear the interrupt flag by reading the combination status register followed by Read Timer MSB. It is also possible, if desired, to change the length of the timer cycle by reinitializing the timer latches. This allows more flexibility for obtaining desired times.

- **Time Interval Modes (TCR3 = 1)**

The Time Interval Modes are provided for applications requiring more flexibility of interrupt generation and Counter Initialization. The Interrupt Flag is set in these modes as a function of both Counter Time Out and transitions of the $\overline{CTG}$ input. Counter Initialization is also affected by Interrupt Flag status. The output signal is not defined in any of these modes. Other features of the Time Interval Modes are Outlined in Table 5.

- **Frequency Comparison Mode (TCR3 = 1, TCR4 = 0)**

The timer within the HD6846 may be programmed to compare the period of a pulse (giving the frequency after calculations) at the $\overline{CTG}$ input with the time period required for Counter Time Out. A negative transition of the $\overline{CTG}$ input enables the counter and starts a Counter Initialization cycle – provided that other conditions as noted in Table 3 are satisfied. The counter decrements on each clock signal recognized during or after Counter Initialization until an Interrupt is generated, a

Table 5 Time Interval Modes

TCR3 = 1			
TCR4	TCR5	APPLICATION	CONDITION FOR SETTING INDIVIDUAL INTERRUPT FLAG
0	0	Frequency Comparison	Interrupt Generated if $\overline{CTG}$ Input Period (1/F) is Less Than Counter Time Out (TO).
0	1	Frequency Comparison	Interrupt Generated if $\overline{CTG}$ Input Period (1/F) is Greater Than Counter Time Out (TO).
1	0	Pulse Width Comparison	Interrupt Generated if $\overline{CTG}$ Input "Down Time" is Less Than Counter Time Out (TO).
1	1	Pulse Width Comparison	Interrupt Generated if $\overline{CTG}$ Input "Down Time" is Greater Than Counter Time Out (TO).

Write Timer Latches command is issued, or a Timer Reset condition occurs. It can be seen from Table 3 that an interrupt condition will be generated if TCR5 = "0" and the period of the pulse (single pulse or measured separately repetative pulses) at the $\overline{CTG}$ input is less than the Counter Time Out period. If TCR5 = "1", an interrupt is generated if the reverse is true.

Assume now with TCR5 = "1" that a Counter Initialization has occurred and that the $\overline{CTG}$ input has returned "Low" prior to Counter Time Out. Since there is no Individual Interrupt Flag generated, this automatically starts a new Counter Initialization Cycle. The process will continue with frequency comparison being performed on each $\overline{CTG}$ input cycle until the mode is changed, or a cycle is determined to be above the predetermined limit.

- **Pulse Width Comparison Mode (TCR3 = 1, TCR4 = 1)**

This mode is similar to the Frequency Comparison Mode except for the limiting factor being a positive, rather than negative, transition of the $\overline{CTG}$ input. With TCR5 = "0", an Individual Interrupt Flag will be generated if the zero level pulse applied to the $\overline{CTG}$ input is less than the time period required for Counter Time Out. With TCR5 = "1", the interrupt is generated when the reverse condition is true.

As can be seen in Table 3, a positive transition of the $\overline{CTG}$ input disables the counter. With TCR5 = "0", it is therefore possible to directly obtain the width of any pulse causing an interrupt.

- **Composite Status Register**

The Composite Status Register (CSR) is a read-only register which is shared by the Timer and the Peripheral Data Port of the HD6846. Three individual interrupt flags in the register are set directly via the appropriate conditions in the timer or peripheral port. The composite interrupt flag – and the $\overline{IRQ}$ Output – respond to these individual interrupts only if corresponding enable bits are set in the appropriate Control Registers. (See Figure 17.) The sequence of assertion is not detected. Setting TCR6 while CSR0 is "High" will cause CSR7 to be set, for example.

The Composite Interrupt Flag (CSR7) is clear only if all enabled Individual Interrupt Flags are clear. The conditions for clearing CSR1 and CSR2 are detailed in a later section. The Timer Interrupt Flag (CSR0) is cleared under the following conditions:

1) Timer Reset – Internal Reset Bit (TCR0) = "1" or External $\overline{RES}$ = "0".
2) Any Counter Initialization condition.
3) A Write Timer Latches command if Time Interval modes (TCR3 = "1") are being used.
4) A Read Timer Counter command, provided this is preceded by a Read Composite Status Register while CSR0 is set. This latter condition prevents missing an Interrupt Request generated after reading the Status Register and prior to reading the counter.

The remaining bits of the Composite Status Register (CSR3~CSR6) are unused. They default to a logic zero when read.

■ I/O OPERATION

- **Parallel Peripheral Port**

The peripheral port of the HD6846 contains 8 Peripheral Data lines (P_0~P_7), two Peripheral Control lines (CP_1 and CP_2), a Data Direction Register, a Peripheral Data Register, and a Peripheral Control Register. The port also directly affects two bits (CSR1 and CSR2) of the Composite Status Register.

The Peripheral Port is similar to the "B" side of a PIA (HD6821) with the following exceptions:

1) All registers are directly accessible in the HD6846 Data Direction and Peripheral Data in the HD6821 are located at the same address with Bit Two of the Control Register used for register selection.
2) Peripheral Control Register Bit Two (PCR2) of the HD6846 is used to select an optional input latch function. This option is not available with HD6821 PIA's.
3) Interrupt Flags are located in the HD6846 composite status register rather than Bits 6 and 7 of the Control Register as used in the HD6821.
4) Interrupt Flags are cleared in the HD6821 by reading data from the Peripheral Data Register. HD6846 Interrupt Flags are cleared by either reading or writing to the Peripheral Data Register – provided that this sequence is followed a) Flag Set, b) Read Composite Status Register, c) Read/Write Peripheral Data Register is followed.

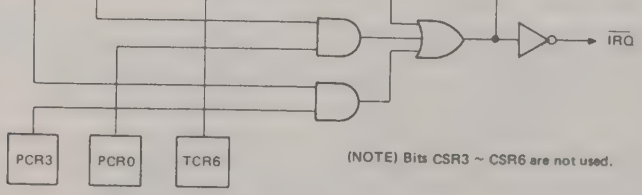


Figure 17 Composite Status Register & Associated Logic

5) Bit 6 of the HD6846 Peripheral Control Register is not used. Bit 7 (PCR7) is an Internal Reset Bit not available on the HD6821.
6) The Peripheral Data lines (and CP_2) of the HD6846 feature internal current limiting which allows them to directly drive the base of Darlington NPN transistors.

- **Data Direction Register**

The MPU can write directly to this eight-bit register to configure the Peripheral Data lines as either inputs or outputs. A particular bit within the register (DDRn) is used to control the corresponding Peripheral Data line (Pn). With DDRn = "0", Pn becomes an input; if DDRn = "1", Pn is an output. As an example, writing Hex $0F into the Data Direction Register results in P_0 thru P_3 becoming outputs and P_4 thru P_7 being inputs. Hex $55 in the Data Direction Register results in alternate outputs and inputs at the parallel port.

- **Peripheral Data Register**

This eight-bit register is used for transferring data between the peripheral data port and the MPU. Any bit corresponding to an output line will be used to drive the output buffer associated with that line. Data in these output bits is normally provided by an MPU Write function. (Input bits – those associated with input lines – are unchanged by a Write Command.) Any input bit will reflect the state of the associated input line if the input latch function is deselected. If the Control Register is programmed to provide input latching, the input bit will retain the state at the time CP_1 was activated until the Peripheral Data Register is read by the MPU.

- **Peripheral Control Register**

This eight-bit register is used to control the reset function as well as for selection of optional functions of the two peripheral control lines (CP_1 and CP_2). The Peripheral Control Register functions are outlined in Table 6.

- **Peripheral Port Reset (PCR7)**

Bit 7 of the Peripheral Control Register (PCR7) may be used to initialize the peripheral section of the HD6846. When this bit is set "High", the peripheral data register, the peripheral data direction register, and the interrupt flags associated with the peripheral port (CSR1 & CSR2) are all cleared. Other bits in the peripheral control register are not affected by PCR7.

PCR7 is set by either a logic zero at the External $\overline{RES}$ input or under program control by writing a "1" into the location. In any case, PCR7 may be cleared only by writing a zero into the location while $\overline{RES}$ is "High". The bit must be cleared to activate the port.

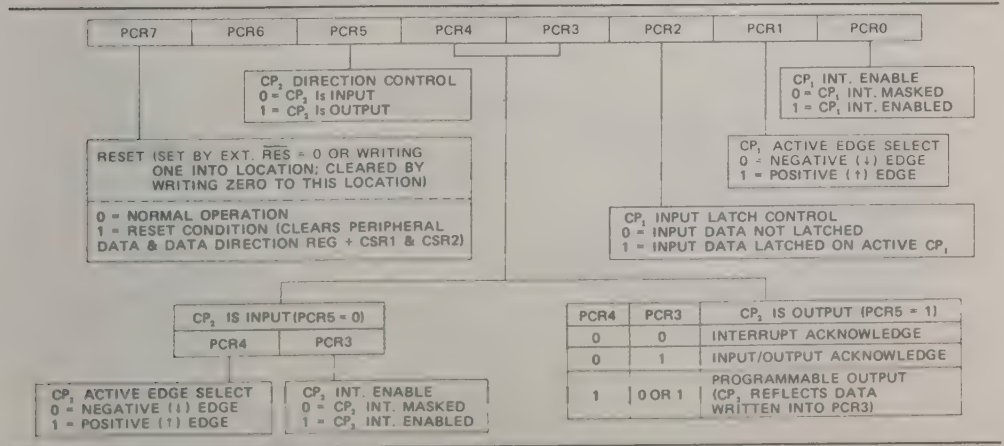
- **Control of CP_1 Peripheral Control Line**

CP_1 may be used an interrupt request to the HD6846, as a strobe to allow latching of input data, or both. In any case, the input can be programmed to be activated by either a positive or negative transition of the signal. Thes options are selected via Control Register Bits PCR0, RCR1 & PCR2.

Control Register Bit 0 (PCR0) is used to enable the interrupt transfer circuitry of the HD6846. Regardless of the state of PCR0, and active transition of CP_1 causes the Composite Status Register Bit One (CSR1) to be set. if PCR0 = "1", this interrupt will be reflected in the Composite Interrupt Flag (CSR7), and thus at the $\overline{IRQ}$ output. CSR1 is cleared by a Peripheral Port Reset condition or by either reading or writing to the peripheral data register after the Composite Status Register is read. The latter alternative is conditional – CSR1 must have been a logic one when the Composite Status Register was lart read. This precludes inadvertent clearing of interrupt flags generated between the time the Status Register is read and the manipulation of peripheral data.

Control Register Bit One (PCR1) is used to select the edge which activates CP_1. When PCR1 = "0", CP_1 is active on negative transitions ("High" to "Low"). "Low" to "High" transitions are sensed by CP_1 when PCR1 = "1".

Table 6 Peripheral Control Register Format (Expanded)



In addition to its use as an interrupt input, CP_1 can be used as a strobe to capture input data in an internal latch. This option is selected by writing a one into Peripheral Control Register Bit Two (PCR2). In operating, the data at the pins designated by the Data Direction Register as inputs will be captured by an active transition of CP_1. An MPU Read of the Peripheral Data Register will result in the captured data being transferred to the MPU – and it also releases the latch to allow capture of new data. Note that successive active transitions with no Read Peripheral Data Command between does not update the input latch. Also, it should be noted that use of the input latch function (which can be deselected by writing a zero into PCR2) has no effect on output data. It also does not affect Interrupt function of CP_1.

- **Control of CP_2 Peripheral Control Line**

CP_2 may be used as an input by writing a zero into PCR5. In this configuration, CP_2 becomes a dual of CP_1 in regard to generation of interrupts. An active transition (as selected by PCR4) causes Bit Two of the Composite Status Register to be set. PCR3 is then used to select whether the CP_2 transition is to cause CSR7 to be set – and thereby cause $\overline{IRQ}$ to go "Low". CP_2 has no effect on the input latch function of the HD6846.

Writing a one into PCR5 causes CP_2 to function as an output. PCR4 then determines whether CP_2 is to be used in a handshake or programmable output mode. With PCR4 = "1", CP_2 will merely reflect the data written into PCR3. Since this can readily be changed under program control, this mode allows CP_2 to be a programmable output line in much the same manner as those lines selected as outputs by the Data Direction Register.

The handshaking mode (PCR5 = "1", PCR4 = "0") allows CP_2 to perform one of two functions as selected by PCR3. With PCR3 = "1", CP_2 will go "Low" on the first Enable positive transition after a Read or Write to the Peripheral Data Register. This Input/Output Acknowledge signal is released (returns "High") on the next positive transition of the Enable signal.

In the Interrupt Acknowledge mode (PCR5 = "1", PCR4 = PCR3 = "0"), CP_2 is set when CSR1 is set by an active transition of CP_1. It is released (goes "Low") on the first positive transition of Enable after CSR1 has been cleared via an MPU Read or Write to the Peripheral Data Register. (Note that the previously described conditions for clearing CSR1 still apply.)

- **Restart Sequence**

A typical restart sequence for the HD6846 will include initialization of both the Peripheral Control & Data Direction Registers of the parallel port. It is necessary to set up the Peripheral Control Register first, since PCR7 = "0" is a condition for writing data into the Data Direction Register. (A logic zero at the external $\overline{RES}$ input automatically sets PCR7.)

- **Summary**

The HD6846 has several optional modes of operation which allow it to be used in a variety of applications. The following tables are provided for reference in selecting these modes.

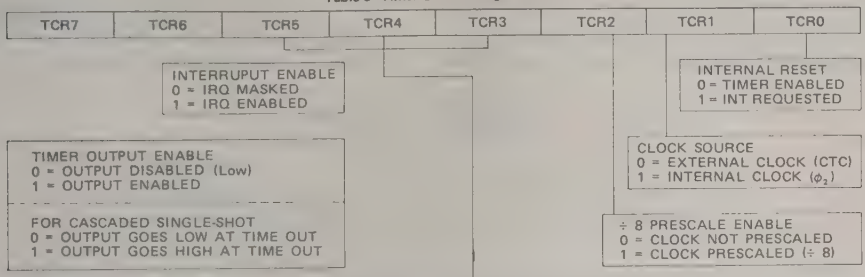
Table 7 HD6846 Internal Register Addresses

R/$\overline{W}$	A_2	A_1	A_0	REGISTER SELECTED
R	0	0	0	Combination Status Register
R/$\overline{W}$	0	0	1	Peripheral Control Register
R/$\overline{W}$	0	1	0	Data Direction Register
R/$\overline{W}$	0	1	1	Peripheral Data Register
R	1	0	0	Combination Status Register
R/$\overline{W}$	1	0	1	Timer Control Register
R/$\overline{W}$	1	1	0	Timer MSB Register
R/$\overline{W}$	1	1	1	Timer LSB Register
R	x	x	x	ROM Address

Table 8 Composite Status Register

CSR7	CSR3~CSR6 NOT USED DEFAULT TO ZERO WHEN READ	CSR2	CSR1	CSR0
COMPOSITE INTERRUPT FLAG 0 = NO ENABLED INTERRUPT FLAG SET 1 = ONE OR MORE ENABLED INTERRUPT FLAGS SET*		CP_2 INTERRUPT FLAG 0 = NO INT REQ 1 = INT REQUESTED		TIMER INTERRUPT FLAG 0 = NO INT REQ. 1 = INT REQUESTED
INVERSE OF THIS BIT APPEARS AT $\overline{IRQ}$ OUTPUT				
*STATUS OF THIS BIT CAN BE EXPRESSED AS: CSR7 = CSR0·TCR6 + CSR1·PCR0 + CSR2·PCR3				CP_1 INTERRUPT FLAG 0 = NO INT REQ. 1 = INT REQUESTED

Table 9 Timer Control Register

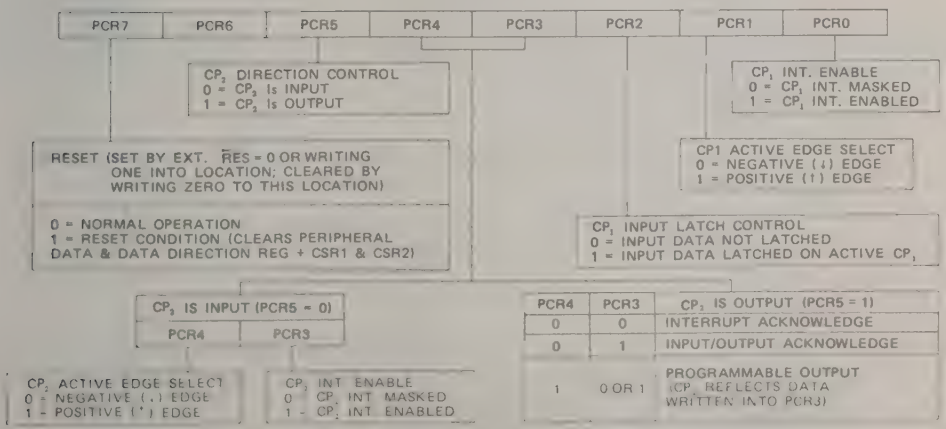


TCR3	TCR4	TCR5	TIME OPERATING MODE	COUNTER INITIALIZATION	INTERRUPT FLAG SET
0	0	0	CONTINUOUS	$\overline{CTG}\downarrow + W + R$	TO
0	0	1	CASCADED SINGLE SHOT	$\overline{CTG}\downarrow + W + R$	TO
0	1	0	CONTINUOUS	$\overline{CTG}\downarrow + R$	TO
0	1	1	NORMAL SINGLE SHOT	$\overline{CTG}\downarrow + R$	TO
1	0	0	FREQUENCY COMPARISON	$\overline{CTG}\downarrow \cdot \overline{I} \cdot (\overline{CE} + TOF \cdot CE) + R$	CTG ↓ BEFORE TO
1	0	1		$\overline{CTG}\downarrow \cdot \overline{I} + R$	TO BEFORE CTG
1	1	0	PULSE WIDTH COMPARISON	$\overline{CTG}\downarrow \cdot \overline{I} + R$	$\overline{CTG}\uparrow$ BEFORE TO
1	1	1		$\overline{CTG}\downarrow \cdot \overline{I} + R$	TO BEFORE $\overline{CTG}\uparrow$

R = RESET CONDITION
W = WRITE TIMER LATCHES
TO = COUNTER TIME OUT
CE = COUNTER ENABLE

$\overline{CTG}\downarrow$ = NEG TRANSISTION OF PIN 17
$\overline{CTG}\uparrow$ = POS TRANSITION OF PIN 17
$\overline{I}$ = INTERRUPT FLAG (CSR0) = 0

Table 10 Peripheral Control Register



■ CUSTOM PROGRAMMING

By the programming of a single photomask for the HD6846, the customer may specify the content of the memory and the method of enabling the outputs.

Information on the general options of the HD6846 should be submitted on an Organizational Data form such as that shown in Figure 18 and Figure 19.

Information for custom memory content may be sent to HITACHI in one of two forms (shown in order of preference):

1) Paper tape output of the HMCS6800 Load Module Format or of the BNPF Format
2) Hexadecimal coding using IBM Punch Cards

ORGANIZATIONAL DATA
HD6846 COMBINATION ROM-I/O-TIMER

Customer:

Company _____

Part No. _____

Originator _____

Phone No. _____

Hitachi Use Only:

Quote:

Part No.:

Specif. No.:

Enable Options: (ROM ENABLE MUST DIFFER FROM I/O-TIMER)

	ROM SECTION		I/O-TIMER SECTION	
	1	0	1	0
CS_0	☐	☐	☐	☐
CS_1	☐	☐	☐	☐

I/O-TIMER SELECT						
A_6			CHECK ONE COLUMN ONLY			
1	0	x				
☐	☐	☐	A_{10}	x	1	x	x	x
			A_9	x	x	1	x	x
			A_8	x	x	x	1	x
			A_7	x	x	x	x	1

1 ≥ 2.0V
0 ≤ 0.8V
x = NOT USED

Figure 18 Format for Programming General Options

DATE

COMPANY ENGINEER SECTION

CUSTOMERS P/N (if you need) ☐☐☐☐☐☐☐☐☐☐☐☐

TYPE NO. OF ROM

DATA FORMAT
1. HMCS6800 load module format 2. BNPF format

coding media
☐ 1. paper tape
☐ 2. IBM 80 column card

total bytes of data (decimal)

initial ROM address (decimal)

parity (for paper tape)
☐ 1. even ☐ 2. odd ☐ 3. none

total number of cards

for HITACHI reference only

ref. No.	
mask ROM No.	
processed data	
approved data	

designed

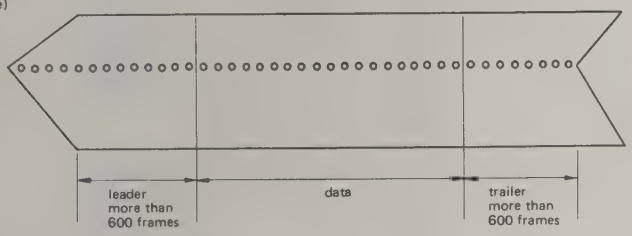
approved

Figure 19 Confirmation sheet of specification for HD6846 series ROM

■ **PAPER TAPE**

1) Any one inch width tape usually available in market can be used but tape in black color is recommended.
2) Both leader and trailer have more than 600 frames.

(Example)



3) One file data of each chip shall be contained in one reel of paper tape. One file data shall not be divided into more than two reels.
4) Parity
   Parity shall be indicated in "Confirmation sheet of specification". Parity forms are grouped;
   (1) With parity EVEN or ODD
   (2) Without parity
5) 8-bit ASCII code shall be used.

■ **CARD**

1) Use IBM 80 column card.
2) Use EBCDIC code.
3) Card format is as follows;
4) Total number of cards shall be written in "Confirmation sheet of specification".

column	contents
1 to 71	Free format of data column
72	Blank
73 to 80	Sequential card number, not free format. Least significant digit of decimal sequential number is located in column 80. No alphabet letters. Any sequential number more than 1 can be used.

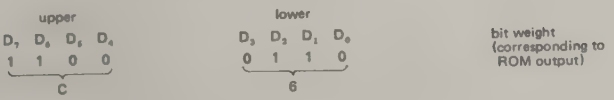
■ **DATA FORMAT**

● **HMCS6800 LOAD MODULE FORMAT**

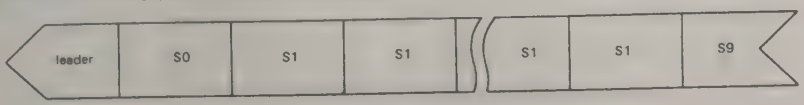
This is object format obtained from HMCS6800 assembler.

1) 8-bit code is divided into upper and lower 4 bits and transformed into hexadecimal number.

(Example) Binary number if 1100 0010 is transformed as follows.

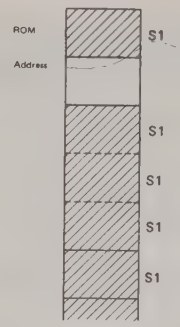


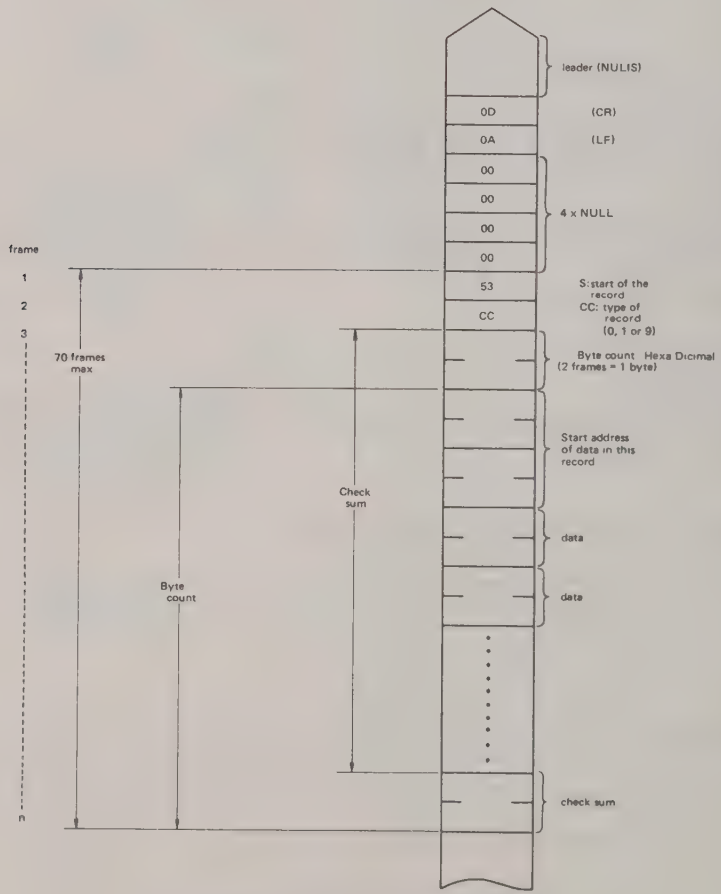
2) Load module structure of paper tape is shown as an example.



S0 is the header record, S1 is data record and S9 is end of file record. Each data record corresponds to each ROM data as shown below. Continuous memory address shall be devided into several records due to limitation of maximum frame number (70 frames = 35 bytes) in one record.

S0, S1 or S9 is distinguished by CC following start of the record S.





Check sum is complement of 1 for sum of each 8-bit.

3) Example of load module format

	frame	CC=30 header record		CC=31 data record		CC=39 end of file record	
1	start of record	53	S	53	S	53	S
2	type of record	30	0	31	1	39	9
3 4	byte count	30 36	06	31 36	16	30 33	03
5 6 7 8	start address of data in this record	30 30 30 30	0000	31 31 30 30	1100	30 30 30 30	0000
9 10	data	34 38	48-H	39 38	98	46 43	FC (check sum)
	data	34 34	44-D	30 32	02		
	data	35 32	52-R				
n	check sum	32 42	2B (check sum)	41 38	A8 (check sum)		

Check sum of header record above is complement of 1 of $(06 + 00 + 00 + 48 + 44 + 52)_{16}$ i.e., 2B.

The start address of data record is incremented for each one byte data, then is compared to the next address in data record and is checked to be sequential or not.

When it is not sequential, hexadecimal 00 is filled as data for that address automatically.

A example of type out of paper tape in HMCS6800 load module format is shown below.

```
header record ... S00600004844522B
data record .....S113F0007EF5587EF7897EFAA77EF9C07EF9C47E24
data record .....S112F010FA657EFA8B7EFAA07EF9DC7EFA247E06
end of file record ....S9030000FC
```

4) Four types of data of ROM code are able to be processed. In any case, header record before data record is needed and so as end of file record after data record.

(a) No vacancy in ROM

Data record is filled with full ROM record of one chip. Therefore address is sequential. Initial ROM address in "Confirmation sheet of specification" is 0.

(b) Vacancy in former part of ROM

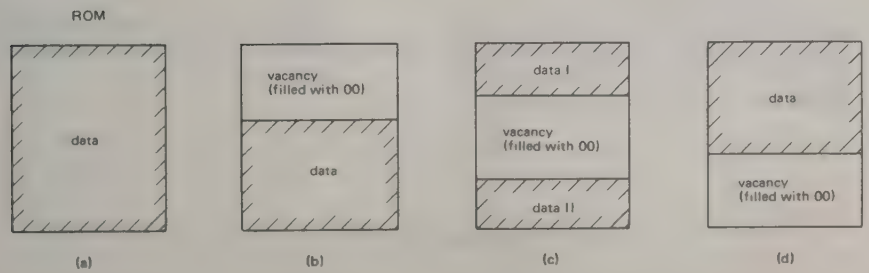
Desired initial address shall be filled in initial ROM address column in "Confirmation sheet of specification". Data of 00 are filled automatically for vacant address.

(c) Vacancy in the middle of ROM

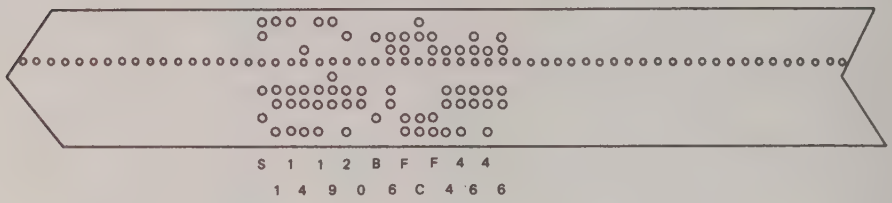
Data of 00 are filled in for vacant address. Initial ROM address for data I is 0 and desired initial address for data II shall be described in "Confirmation sheet of specification".

(d) Vacancy in later part of ROM

When end of file record is read out, data of 00 are filled in thereafter.



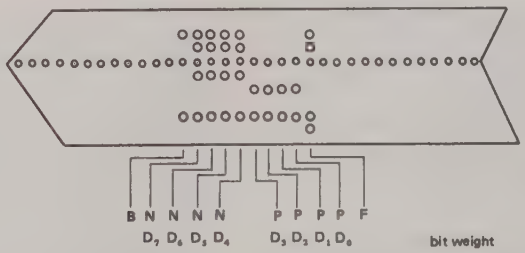
(Example) Paper tape whose data record is S1141920B6FC .....



- **BNPF format**

1) Each word is expressed as BNPF slice which begins word opening mark B, has 8 character bit contents shown by P or N and finishes with end mark F.

(Example) 0F in hexadecimal code is expressed as shown below (paper tape).



2) Any contents between F of the first slice and B of next slice are disregarded.
3) Bit pattern (BNPF) slice for all ROM address shall be indicated. Initial ROM address in "Confirmation sheet of specification" is, therefore always 0 for BNPF.
   - B ..... shows beginning of the word
   - N ..... shows 0 of one bit data
   - P ..... shows 1 of one bit data
   - F ..... shows end of the word

Note 1) X can be used expect for P and N for indication of word contents of BNPF slice. This X means that bit can be either P or N (don't care). X shall be determined by HITACHI for testing and shall be informed to the customer in confirmation table.

Note 2) Expression of B*nF can be used for indicating that the same contents of foregoing slice are applicable from this word to following n words.
For example, when B*4F is indicated at 10th word position, the contents of 9th word are repeated for 10, 11, 12 and 13th word.
(Content of X is not always repeated even in this case.)
n is grater than 1 and less than final address of ROM.

Note 3) When vacancy of ROM exists, combination of Note 1) and Note 2) is usefull.

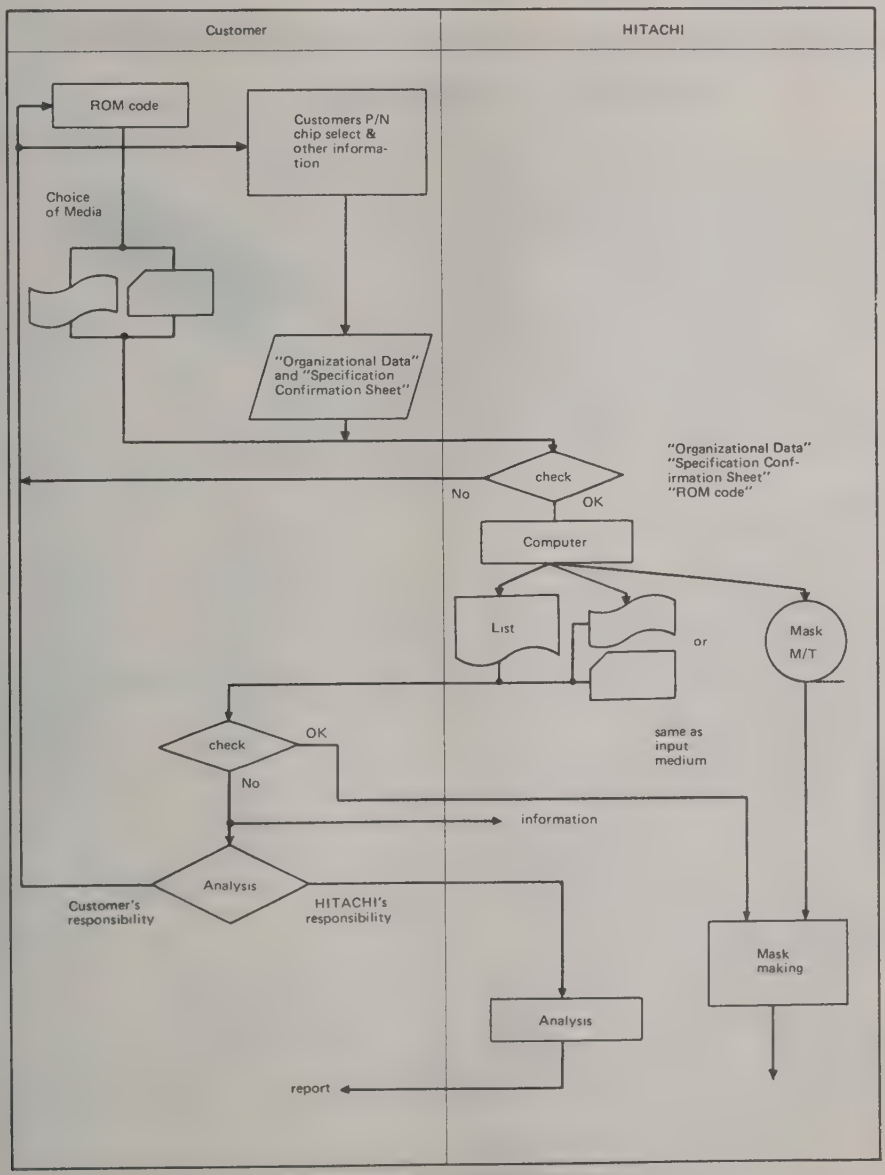


Figure 20 Flow chart of Mask ROM Development

HD6850, HD68A50

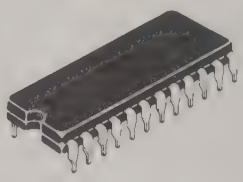
ACIA (Asynchronous Communications Interface Adapter)

The HD6850 Asynchronous Communications Interface Adapter provides the data formatting and control to interface serial asynchronous data communications information to bus organized systems such as the HMCS6800 Microprocessing Unit.

The bus interface of the HD6850 includes select, enable, read/write, interrupt and bus interface logic to allow data transfer over an 8-bit bi-directional data bus. The parallel data of the bus system is serially transmitted and received by the asynchronous data interface, with proper formatting and error checking.

The functional configuration of the ACIA is programmed via the data bus during system initialization. A programmable Control Register provides variable word lengths, clock division ratios, transmit control, receive control, and interrupt control. For peripheral or modem operation three control lines are provided.

HD6850P, HD68A50P

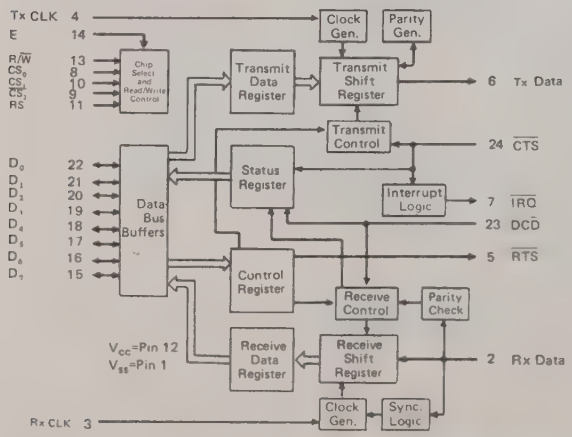


(DP-24)

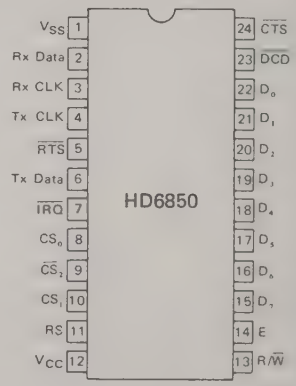
■ FEATURES

- Serial/Parallel Conversion of Data
- Eight and Nine-bit Transmission
- Insertion and Deleting of Start and Stop Bit
- Optional Even and Odd Parity
- Parity, Overrun and Framing Error Checking
- Peripheral/Modem Control Functions (Clear to Send $\overline{CTS}$, Request to Send $\overline{RTS}$, Data Carier Detect $\overline{DCD}$)
- Optional ÷ 1, ÷ 16, and ÷ 64 Clock Modes
- Up to 500kbps Transmission
- Programmable Control Register
- N-channel Silicon Gate Process
- Compatible with MC6850 and MC68A50

■ BLOCK DIAGRAM



■ PIN ARRANGEMENT



(Top View)

HITACHI

■ ABSOLUTE MAXIMUM RATINGS

Item	Symbol	Value	Unit
Supply Voltage	V_{CC}*	-0.3 ~ +7.0	V
Input Voltage	V_{in}*	-0.3 ~ +7.0	V
Operating Temperature	T_{opr}	-20 ~ +75	°C
Storage Temperature	T_{stg}	-55 ~ +150	°C

* With respect to V_{SS} (SYSTEM GND)

(NOTE) Permanent LSI damage may occur if maximum ratings are exceeded. Normal operation should be under recommended operating conditions. If these conditions are exceeded, it could affect reliability of LSI.

■ RECOMMENDED OPERATING CONDITIONS

Item	Symbol	min	typ	max	Unit
Supply Voltage	V_{CC}*	4.75	5.0	5.25	V
Input Voltage	V_{IL}*	-0.3	–	0.8	V
	V_{IH}*	2.0	–	V_{CC}	V
Operating Temperature	T_{opr}	-20	25	75	°C

* With respect to V_{SS} (SYSTEM GND)

■ ELECTRICAL CHARACTERISTICS

● DC CHARACTERISTICS (V_{CC}=5V±5%, V_{SS}=0V, Ta=-20~+75°C, unless otherwise noted.)

Item		Symbol	Test Condition	min	typ*	max	Unit
Input "High" Voltage	All Inputs	V_{IH}		2.0	–	V_{CC}	V
Input "Low" Voltage	All Inputs	V_{IL}		-0.3	–	0.8	V
Input Leakage Current	R/$\overline{W}$, CS_0, CS_1, $\overline{CS_2}$, E	I_{in}	V_{in}=0~5.25V	-2.5	–	2.5	μA
Three-State (Off State) Input Current	D_0~D_7	I_{TSI}	V_{in}=0.4~2.4V	-10	–	10	μA
Output "High" Voltage	D_0~D_7	V_{OH}	I_{OH}=-205μA, Enable Pulse Width ≦ 25μs	2.4	–	–	V
	TxData, $\overline{RTS}$		I_{OH}=-100μA, Enable Pulse Width ≦ 25μs	2.4	–	–	
Output "Low" Voltage	All outputs	V_{OL}	I_{OL}=1.6mA, Enable Pulse Width ≦ 25μs	–	–	0.4	V
Output Leakage Current (Off State)	$\overline{IRQ}$	I_{LOH}	V_{OH}=2.4V	–	–	10	μA
Power Dissipation		P_D		–	300	525	mW
Input Capacitance	D_0~D_7	C_{in}	V_{in}=0V, T_a=25°C, f=1.0MHz	–	–	12.5	pF
	E, TxCLK, RxCLK, R/$\overline{W}$, RS, RxData, CS_0, CS_1, $\overline{CS_2}$, $\overline{CTS}$, $\overline{DCD}$			–	–	7.5	
Output Capacitance	$\overline{RTS}$, TxData	C_{out}	V_{in}=0V, T_a=25°C, f=1.0MHz	–	–	10	pF
	$\overline{IRQ}$			–	–	5.0	

* T_a=25°C, V_{CC}=5V

• AC CHARACTERISTICS

1. TIMING OF DATA TRANSMISSION

Item		Symbol	Test Condition	min	typ	max	Unit
Minimum Clock Pulse Width	÷16, ÷64 Modes	PW_{CL}	Fig. 1	600	—	—	ns
	÷16, ÷64 Modes	PW_{CH}	Fig. 2	600	—	—	ns
Clock Frequency	÷1 Mode	f_c		—	—	500	kHz
	÷16, ÷64 Modes			—	—	800	
Clock-to-Data Delay for Transmitter		t_{TDD}	Fig. 3	—	—	1.0	μs
Receive Data Setup Time	÷ 1 Mode	t_{RDSU}	Fig. 4	500	—	—	ns
Receive Data Hold Time	÷ 1 Mode	t_{RDH}	Fig. 5	500	—	—	ns
$\overline{IRQ}$ Release Time		t_{IR}	Fig. 6	—	—	1.2	μs
$\overline{RTS}$ Delay Time		t_{RTS}	Fig. 6	—	—	1.0	μs
Rise Time and Fall Time	Except E	t_r, t_f		—	—	1.0*	μs

* 1.0 μs or 10% of the pulse width, whichever is smaller.

2. BUS TIMING CHARACTERISTICS

1) READ

Item	Symbol	Test Condition	HD6850			HD68A50			Unit
			min	typ	max	min	typ	max	
Enable Cycle Time	t_{cycE}	Fig. 7	1.0	—	—	0.666	—	—	μs
Enable "High" Pulse Width	PW_{EH}	Fig. 7	0.45	—	25	0.28	—	25	μs
Enable "Low" Pulse Width	PW_{EL}	Fig. 7	0.43	—	—	0.28	—	—	μs
Setup Time, Address and R/$\overline{W}$ valid to Enable positive transition	t_{AS}	Fig. 7	140	—	—	140	—	—	ns
Data Delay Time	t_{DDR}	Fig. 7	—	—	320	—	—	220	ns
Data Hold Time	t_H	Fig. 7	10	—	—	10	—	—	ns
Address Hold Time	t_{AH}	Fig. 7	10	—	—	10	—	—	ns
Rise and Fall Time for Enable Input	t_{Er}, t_{Ef}	Fig. 7	—	—	25	—	—	25	ns

2) WRITE

Item	Symbol	Test Condition	HD6850			HD68A50			Unit
			min	typ	max	min	typ	max	
Enable Cycle Time	t_{cycE}	Fig. 8	1.0	—	—	0.666	—	—	μs
Enable "High" Pulse Width	PW_{EH}	Fig. 8	0.45	—	25	0.28	—	25	μs
Enable "Low" Pulse Width	PW_{EL}	Fig. 8	0.43	—	—	0.28	—	—	μs
Setup Time, Address and R/$\overline{W}$ valid to Enable positive transition	t_{AS}	Fig. 8	140	—	—	140	—	—	ns
Data Setup Time	t_{DSW}	Fig. 8	195	—	—	80	—	—	ns
Data Hold Time	t_H	Fig. 8	10	—	—	10	—	—	ns
Address Hold Time	t_{AH}	Fig. 8	10	—	—	10	—	—	ns
Rise and Fall Time for Enable Input	t_{Er}, t_{Ef}	Fig. 8	—	—	25	—	—	25	ns

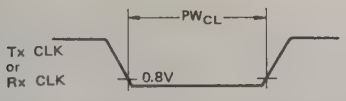


Figure 1 Clock Pulse Width, "Low" State

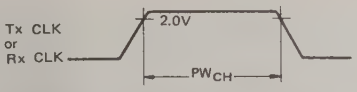


Figure 2 Clock Pulse Width, "High" State

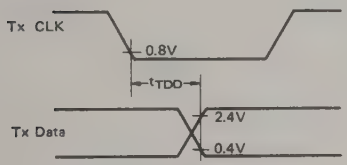


Figure 3 Transmit Data Output Delay

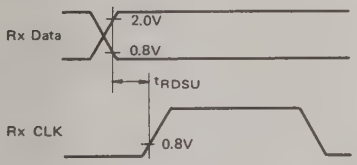


Figure 4 Receive Data Setup Time (÷1 Mode)

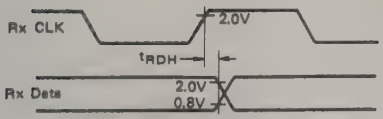
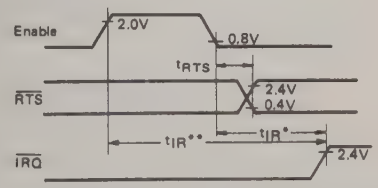


Figure 5 Receive Data Hold Time (÷1 Mode)



* (1) $\overline{IRQ}$ Release Time applied to Rx Data Register read operation.
(2) $\overline{IRQ}$ Release Time applied to Tx Data Register write operation.
(3) $\overline{IRQ}$ Release Time applied to control Register write TIE = 0, RIE = 0 operation.

** $\overline{IRQ}$ Release Time applied to Rx Data Register read operation right after read status register, when $\overline{IRQ}$ is asserted by $\overline{DCD}$ rising edge.

(Note) Note that followings take place when $\overline{IRQ}$ is asserted by the detection of transmit data register empty status. $\overline{IRQ}$ is released to "High" asynchronously with E signal when $\overline{CTS}$ goes "High". (Refer to Figure 14)

Figure 6 $\overline{RTS}$ Delay and $\overline{IRQ}$ Release Time

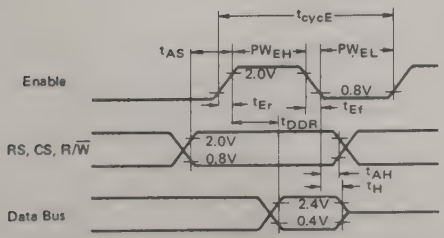


Figure 7 Bus Read Timing Characteristics
(Read information from ACIA)

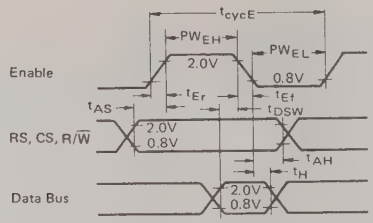
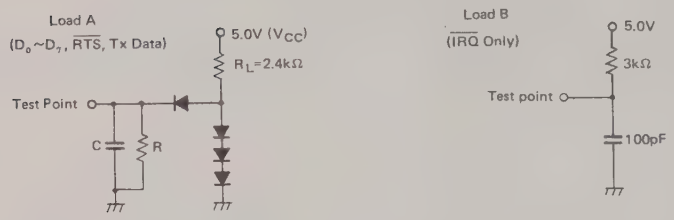


Figure 8 Bus Write Timing Characteristics
(Write information into ACIA)



C = 130pF for D_0~D_7
= 30pF for $\overline{RTS}$ and Tx Data
All diodes are 1S2074 Ⓗ or Equivalent.

R = 11kΩ for D_0~D_7
= 24kΩ for $\overline{RTS}$ and Tx Data

Figure 9 Bus Timing Test Loads

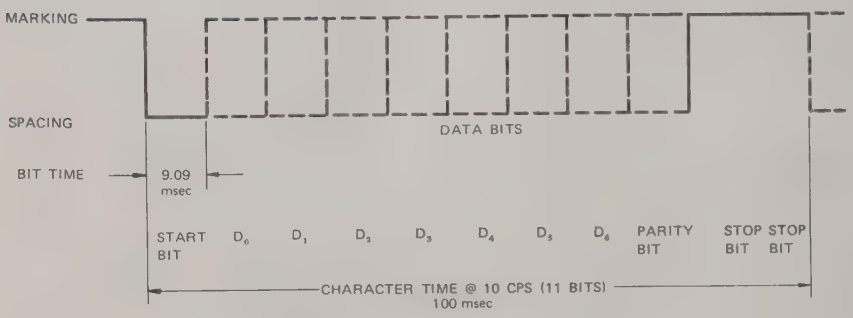
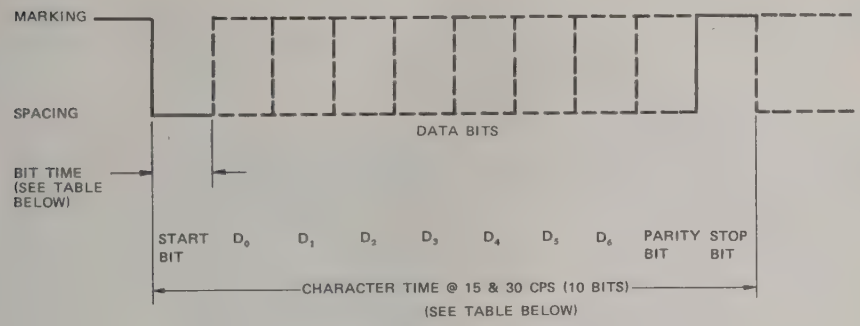


Figure 10 110 Baud Serial ASCII Data Timing



BAUD RATE	150	300
CHARACTERS/SEC	15	30
BIT TIME (msec)	6.67	3.33
CHARACTER TIME (msec)	66.7	33.3

$$\text{BIT TIME} = \frac{\text{SEC}}{\text{BAUD RATE}}$$

Figure 11 150 & 300 Baud Serial ASCII Data Timing

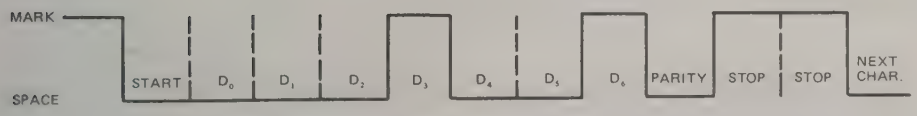


Figure 12 Send a 7 Bit ASCII Char. "H" Even Parity – 2 Stop Bits $H = 48_{16} = 1001000_2$

■ DATA OF ACIA

HD6850 is an interface adapter which controls transmission and reception of Asynchronous serial data. Some examples of serial data are shown in Figs. 10 ~ 12.

■ INTERNAL STRUCTURE OF ACIA

HD6850(ACIA) provides the following; 8-bit Bi-directional Data Buses (D_0~D_7), Receive Data Input (Rx Data), Transmit Data Output (Tx Data), three Chip Selects (CS_0, CS_1, $\overline{CS_2}$), Register Select Input (RS), Two Control Input (Read/Write (R/$\overline{W}$), Enable(E) , Interrupt Request Output($\overline{IRQ}$), Clear-to-Send ($\overline{CTS}$) to control the modem, Request-to-Send ($\overline{RTS}$), Data Carrier Detect($\overline{DCD}$) and Clock Inputs(Tx CLK, Rx CLK) used for synchronization of received and transmitted data. This ACIA also provides four registers; Status Register, Control Register, Receive Register and Transmit Register.

24-pin dual-in-line type package is used for the ACIA. Internal Structure of ACIA is illustrated in Fig. 13.

■ ACIA OPERATION

● Master Reset

ACIA has an internal master reset function controlled by software, since it has no hardware reset pin. Bit 0 and bit 1 of control register should be set to "11" to execute master reset, also bit 5 and bit 6 should be programmed to get predetermined $\overline{RTS}$ output accordingly. To release the master reset, the data other than "11" should be written into bit 0, bit 1 of the control register. When the master reset is released, the control register needs to be programmed to get predetermined options such as clock divider ratios, word length, one or two stop bits, parity (even, old, or none), etc.

It may happen that "Low" level output is provided in $\overline{IRQ}$ pin during the time after power-on till master reset. In the system using ACIA, interrupt mask bit of MPU should be released after the master reset of ACIA. (MPU interrupt should be prohibited until MPU program completes the master reset of ACIA.) Transmit Data Register (TDR) and Receive Data Register (RDR) can not be reset by master reset.

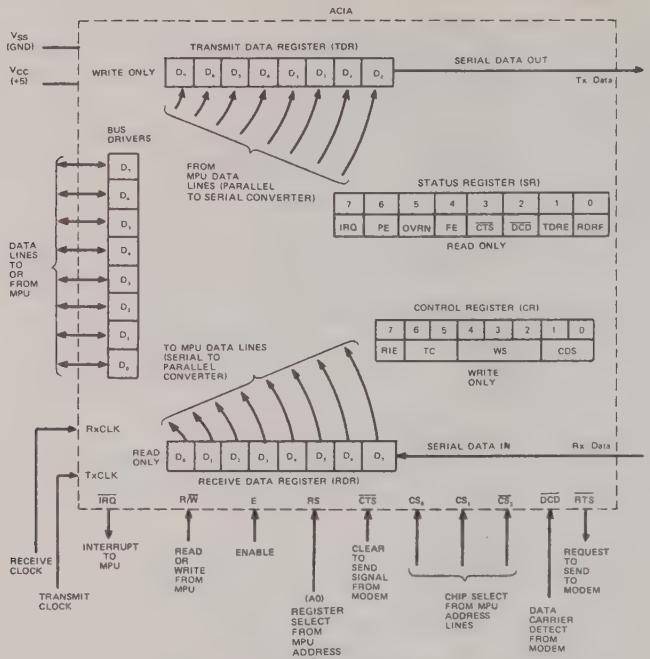


Figure 13 Internal Structure of ACIA

- **Transmit**

A typical transmitting sequence consists of reading the ACIA Status Register either as a result of an interrupt or in the ACIA's turn in a polling sequence. A character may be written into the Transmit Data Register if the status read operation has indicated that the Transmit Data Register is empty. This character is transferred to a Shift Register where it is serialized and transmitted from the Transmit Data output preceded by a start bit and followed by one or two stop bits. Internal parity (odd or even) can be optionally added to the character and will occur between the last data bit and the first stop bit. After the first character is written in the Data Register, the Status Register can be read again to check for a Transmit Data Register Empty condition and current peripheral status. If the register is empty, another character can be loaded for transmission even though the first character is in the process of being transmitted (because of double buffering). The second character will be automatically transferred into the Shift Register when the first character transmission is completed. This sequence continues until all the characters have been transmitted.

- **Receive**

Data is received from a peripheral by means of the Receive Data input. A divide-by-one clock ratio is provided for an externally synchronized clock (to its data) while the divide-by-16 and 64 ratios are provided for internal synchronization. Bit synchronization in the divide-by-16 and 64 modes is initiated by the detection of the leading mark-space transition of the start bit. False start bit detection capability insures that a full half bit of a start bit has been received before the internal clock is synchronized to the bit time. As a character is being received, parity (odd or even) will be checked and the error indication will be available in the Status Register along with framing error, overrun error, and Receive Data Register full. In a typical receiving sequence, the Status Register is read to determine if a character has been received from a peripheral. If the Receiver Data Register is full, the character is placed on the 8-bit ACIA bus when a Read Data command is received from the MPU. When parity has been selected for an 8-bit word (7 bits plus parity), the receiver strip the parity bit (D_7="0") so that data alone is transferred to the MPU. This feature reduces MPU programming. The Status Register can continue to be read again to determine when another character is available in the Receive Data Register. The receiver is also double buffered so that a character can be read from the data register as another character is being received in the Shift register. The above sequence continues until all characters have been received.

■ ACIA INTERNAL REGISTERS

The ACIA provides four registers; Transmit Data Register (TDR), Receive Data Register(RDR), Control Register(CR) and Status Register(SR). The content of each of the registers is summarized in Table 1.

Table 1 Definition of ACIA Register Contents

Buffer Address	RS=1 · R/$\overline{W}$=0 ****	RS=1 · R/$\overline{W}$=1	RS=0 · R/$\overline{W}$=0	RS=0 · R/$\overline{W}$=1
Data Bus	Transmit Data Register	Receiver Data Register	Control Register	Status Register
	(Write Only)	(Read Only)	(Write Only)	(Read Only)
0	Data Bit 0*	Data Bit 0	Counter Divide Select (CR0)	Rx Data Reg. Full (RDRF)
1	Data Bit 1	Data Bit 1	Counter Divide Select (CR1)	Tx Data Reg. Empty (TDRE)
2	Data Bit 2	Data Bit 2	Word Select 1 (CR2)	Data Carrier Detect ($\overline{DCD}$)
3	Data Bit 3	Data Bit 3	Word Select 2 (CR3)	Clear to Send ($\overline{CTS}$)
4	Data Bit 4	Data Bit 4	Word Select 3 (CR4)	Framing Error (FE)
5	Data Bit 5	Data Bit 5	Tx Control 1 (CR5)	Overrun (OVRN)
6	Data Bit 6	Data Bit 6	Tx Control 2 (CR6)	Parity Error (PE)
7	Data Bit 7***	Data Bit 7**	Rx Interrupt Enable (CR7)	Interrupt Request (IRQ)

* Leading bit = LSB = Bit 0
** Data bit will be zero in 7-bit plus parity modes.
*** Data bit is "don't care" in 7-bit plus parity modes.
**** 1 ... "High" level, 0 ... "Low" level

- **Transmit Data Register (TDR)**

Data is written in the Transmit Data Register during the negative transition of the enable (E) when the ACIA has been addressed and RS · R/$\overline{W}$ is selected. Writing data into the register causes the Transmit Data Register Empty bit in the Status Register to go "0". Data can then be transmitted. If the transmitter is idling and no character is being transmitted, then the transfer will take place within 2 bit time + several E cycles of the trailing edge of the Write command. If a character is being transmitted, the new data character will commence as soon as the previous character is complete. The transfer of data causes the Transmit Data Register Empty (TDRE) bit to indicate empty.

- **Receive Data Register (RDR)**

Data is automatically transferred to the empty Receive Data Register (RDR) from the receiver deserializer (a shift register) upon receiving a complete character. This event causes the Receive Data Register Full bit (RDRF) on the status buffer to go "1" (full). Data may then be read through the bus by addressing the ACIA and R/$\overline{W}$ "High" when the ACIA is enabled. The non-destructive read cycle causes the RDRF bit to be cleared to empty although the data is retained in the RDR. The status is maintained by RDRF as to whether or not the data is current. When the Receive Data Register is full, the automatic transfer of data from the Receiver Shift Register to the Data Register is inhibited and the RDR contents remain valid with its current status stored in the Status Register.

- **Control Register**

The ACIA Control Register consists of eight bits of write-only buffer that are selected when RS and R/$\overline{W}$ are "Low". This register controls the function of the receiver, transmitter, interrupt enables, and the Request-to-Send ($\overline{RTS}$) peripheral/modem control output.

Counter Divide Select Bits (CR0 and CR1)

The Counter Divide Select Bits (CR0 and CR1) determine the divide ratios utilized in both the transmitter and receiver section of the ACIA. Additionally, these bits are used to provide a master reset for the ACIA which clears the Status Register (except for external conditions on $\overline{CTS}$ and $\overline{DCD}$) and initializes both the receiver and transmitter. Master reset does not affect other Control Register bits. Note that after power-on or a power fail/restart, these bits must be set "1" to reset the ACIA. After resetting, the clock divide ratio may be selected. These counter select bits provide for the following clock divide ratios:

Table 2 Function of Counter Devide Select Bit

CR1	CR0	Function
0	0	÷1
0	1	÷16
1	0	÷64
1	1	Master Reset

Word Select Bits (CR2, CR3, and CR4)

The Word Select bits are used to select word length, parity, and the number of stop bits. The encoding format is as follows:

Table 3 Function of Word Select Bit

CR4	CR3	CR2	Function
0	0	0	7 Bits + Even Parity + 2 Stop Bits
0	0	1	7 Bits + Odd Parity + 2 Stop Bits
0	1	0	7 Bits + Even Parity + 1 Stop Bit
0	1	1	7 Bits + Odd Parity + 1 Stop Bit
1	0	0	8 Bits + 2 Stop Bits
1	0	1	8 Bits + 1 Stop Bit
1	1	0	8 Bits + Even Parity + 1 Stop Bit
1	1	1	8 Bits + Odd Parity + 1 Stop Bit

Word length, Parity Select, and Stop Bit changes are not buffered and therefore become effective immediately.

Transmitter Control Bits (CR5 and CR6)

Two Transmitter Control bits provide for the control of the interrupt from the Transmit Data Register Empty condition, the Request-to-Send ($\overline{RTS}$) output, and the transmission of a Break level (space). The following encoding format is used:

Table 4 Function of Transmitter Control-Bit

CR6	CR5	Function
0	0	$\overline{RTS}$ = "Low", Transmitting Interrupt Disabled.
0	1	$\overline{RTS}$ = "Low", Transmitting Interrupt Enabled.
1	0	$\overline{RTS}$ = "High", Transmitting Interrupt Disabled.
1	1	$\overline{RTS}$ = "Low", Transmits a Break level on the Transmit Data Output. Transmitting Interrupt Disabled.

Receive Interrupt Enable Bit (CR7)

The following interrupts will be enabled by a "1" in bit position 7 of the Control Register (CR7): Receive Data Register Full, Overrun, or a "Low" to "High" transistion on the Data Carrier Detect ($\overline{DCD}$) signal line.

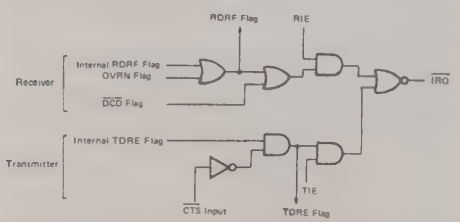


Figure 14 $\overline{IRQ}$ Internal Circuit

- **Status Register**

Information on the status of the ACIA is available to the MPU by reading the ACIA Status Register. This read-only register is selected when RS is "Low" and R/$\overline{W}$ is "High". Information stored in this register indicates the status of the Transmit Data Register, the Receive Data Register and error logic, and the peripheral/modem status inputs of the ACIA.

Receive Data Register Full (RDRF), Bit 0

RDRF indicates that received data has been transferred to the Receive Data Register. RDRF is cleared after an MPU read of the Receive Data Register or by a master reset. The cleared or empty state indicates that the contents of the Receive Data Register are not current. Data Carrier Detect ($\overline{DCD}$) being "High" also causes RDRF to indicate empty.

Transmit Data Register Empty (TDRE), Bit 1

The Transmit Data Register Empty bit being set "1" indicates that the Transmit Data Register contents have been transferred and that new data may be entered. The "0" state indicates that the register is full and that transmission of a new character has not begun since the last write data command.

Data Carrier Detect ($\overline{DCD}$), Bit 2

The $\overline{DCD}$ bit will be "1" when the $\overline{DCD}$ input from a modem has gone "High" to indicate that a carrier is not present. This bit going "1" causes an Interrupt Request to be generated when the Receive Interrupt Enable is set. It remains "1" after the $\overline{DCD}$ input is returned "Low" until cleared by first reading the Status Register and then the Data Register or until a master reset occurs. If the $\overline{DCD}$ input remains "High" after read status and read data or master reset has occurred, the interrupt is cleared, the $\overline{DCD}$ status bit remains "1" and will follow the $\overline{DCD}$ input.

Clear-to-Send ($\overline{CTS}$), Bit 3

The $\overline{CTS}$ bit indicates the state of the $\overline{CTS}$ input from a modem. A "Low" $\overline{CTS}$ indicates that there is a $\overline{CTS}$ from the modem. In the "High state, the Transmit Data Register Empty bit is inhibited and the $\overline{CTS}$ status bit will be "1". Master reset does not affect the Clear-to-Send Status bit.

Framing Error (FE), Bit 4

FE indicates that the received character is improperly framed by a start and a stop bit and is detected by the absence of the 1st stop bit. This error indicates a synchronization error, faulty transmission, or a break condition. The FE flag is set or reset during the receive data transfer time. Therefore, this error indicator is present throughout the time that the associated character is available.

Receiver Overrun (OVRN), Bit 5

Overrun is an error flag that indicates that one or more characters in the data stream were lost. That is, a character or a number of characters were received but not read from the Receive Data Register (RDR) prior to subsequent characters being received. The overrun condition begins at the midpoint of the last bit of the second character received in succession without a read of the RDR having occurred. The overrun does not occur in the Status Register until the valid character prior to Overrun has been read. The RDRF bit remains set until the Overrun is reset. Character synchronization is maintained during the Overrun condition. The Overrun indication is reset after the reading of data from the Receive Data Register or by a Master Reset.

Parity Error (PE), Bit 6

The PE flag indicates that the number of "1"s (highs) in the character does not agree with the preselected odd or even parity. Odd parity is defined to be when the total number of ones is odd. The parity error indication will be present as long as the data character is in the RDR. If no parity is selected, then both the transmitter parity generator output and the receiver parity check results are inhibited.

Interrupt Request (IRQ), Bit 7

The IRQ bit indicates the state of the $\overline{IRQ}$ output. Any interrupt condition with its applicable enable will be indicated in this status bit. Anytime the $\overline{IRQ}$ output is "Low" the IRQ bit will be "1" to indicate the interrupt or service request status. IRQ is cleared by a read operation to the Receive Data Register or a write operation to the Transmit Data Register. (Refer to Figure 14.)

■ SIGNAL FUNCTIONS

● Interface Signal for MPU

Bi-Directional Data Bus (D_0~D_7)

The bi-directional data bus (D_0~D_7) allow for data transfer between the ACIA and the MPU. The data bus output drivers are three-state devices that remain in the high impedance (off) state except when the MPU performs an ACIA read operation.

Enable (E)

The Enable signal, E, is a high impedance TTL compatible input that enables the bus input/output data buffers and clocks data to and from the ACIA. This signal will normally be a derivative of the HMCS6800 ϕ_2 Clock.

Read/Write (R/$\overline{W}$)

The R/$\overline{W}$ line is a high impedance input that is TTL compatible and is used to control the direction of data flow through the ACIA's input/output data bus interface. When R/$\overline{W}$ is "High" (MPU Read cycle), ACIA output drivers are turned on and a selected register is read. When it is "Low", the ACIA output drivers are turned off and the MPU writes into a selected register. Therefore, the R/$\overline{W}$ signal is used to select read-only or write-only registers within the ACIA.

Chip Select (CS_0, CS_1, $\overline{CS_2}$)

These three high impedance TTL compatible input lines are used to address the ACIA. The ACIA is selected when CS_0 and CS_1 are "High" and $\overline{CS_2}$ is "Low". Transfers of data to and from the ACIA are then performed under the control of the Enable signal, Read/Write, and Register Select.

Register Select (RS)

The RS line is a high impedance input that is TTL compatible. A "High" level is used to select the Transmit/Receive Data Registers and a "Low" level the Control/Status Registers. The R/$\overline{W}$ signal line is used in conjunction with Register Select to select the read-only or write-only register in each register pair.

Interrupt Request ($\overline{IRQ}$)

$\overline{IRQ}$ is a TTL compatible, open-drain (no internal pullup), active "Low" output that is used to interrupt the MPU. The $\overline{IRQ}$ output remains "Low" as long as the cause of the interrupt is present and the appropriate interrupt enable within the ACIA is set.

Clock Inputs

Separate high impedance TTL compatible inputs are provided for clocking of transmitted and received data. Clock frequencies of 1, 16 or 64 times the data rate may be selected.

Transmit Clock (Tx CLK)

The Tx CLK input is used for the clocking of transmitted data. The transmitter initiates data on the negative transition of the clock.

Receive Clock (Rx CLK)

The Rx CLK input is used for synchronization of received data. (In the ÷ 1 mode, the clock and data must be synchronized externally.) The receiver samples the data on the positive transition of the clock.

● Serial Input/Output Lines

Receive Data (Rx Data)

The Rx Data line is a high impedance TTL compatible input through which data is received in a serial format. Synchronization with a clock for detection of data is accomplished internally when clock rates of 16 or 64 times the bit rate are used. Data rates are in the range of 0 to 500 kbps when external synchronization is utilized.

Transmit Data (Tx Data)

The Tx Data output line transfers serial data to a modem or other peripheral. Data rates in the range of 0 to 500 kbps when external synchronization is utilized.

Modem Control

The ACIA includes several functions that permit limited control of a peripheral or modem. The functions included are $\overline{CTS}$, $\overline{RTS}$ and $\overline{DCD}$.

Clear-to-Send ($\overline{CTS}$)

This high impedance TTL compatible input provides automatic control of the transmitting end of a communications link via the modem $\overline{CTS}$ active "Low" output by inhibiting the Transmit Data Register Empty (TDRE) status bit. (Refer to Figure 15.)

Request-to-Send ($\overline{RTS}$)

The $\overline{RTS}$ output enables the MPU to control a peripheral or modem via the data bus. The $\overline{RTS}$ output corresponds to the state of the Control Register bits CR5 and CR6. When CR6=0 or both CR5 and CR6=1, the $\overline{RTS}$ output is "Low" (the active state). This output can also be used for Data Terminal Ready ($\overline{DTR}$). (Refer to Figure 15.)

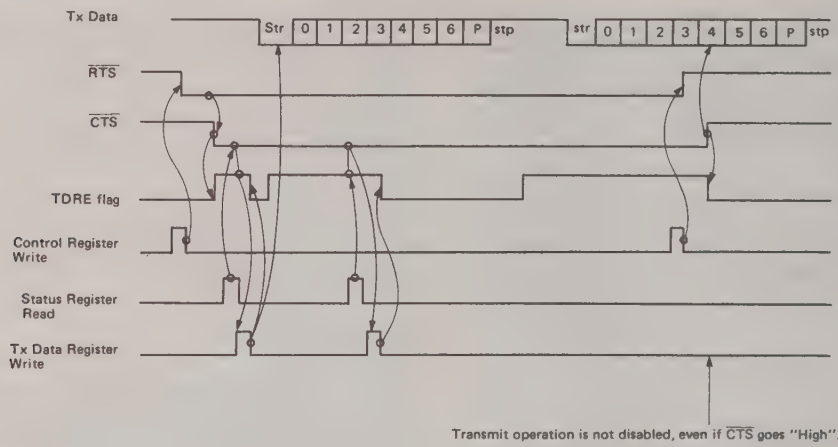


Figure 15 RTS and CTS Timing Chart (Example of 2 bytes transmission)

Data Carrier Detect (DCD)

DCD is the input signal corresponding to the "carrier detect" signal which shows carrier detect of modem.

DCD signal is used to control the receiving operation. When DCD input goes "High", ACIA stops all the receiving operation and sets receiving part in reset status. It means that receive shift register stops shifting, error detection circuit and synchronization circuit of receive clock are reset. When DCD is in "High" level, the receiving part of ACIA is kept in initial status and the operation in the receiving part is prohibited. When DCD goes "Low", the receiving part is allowed to receive data. In this case, the following process is needed to reset DCD flag and restarts the receive operation. (Refer to Figure 16.)

(1) Return DCD input from "High" to "Low".
(2) Read status register. (DCD flag = "1")
(3) Read receive data register (Uncertain data will be read.)

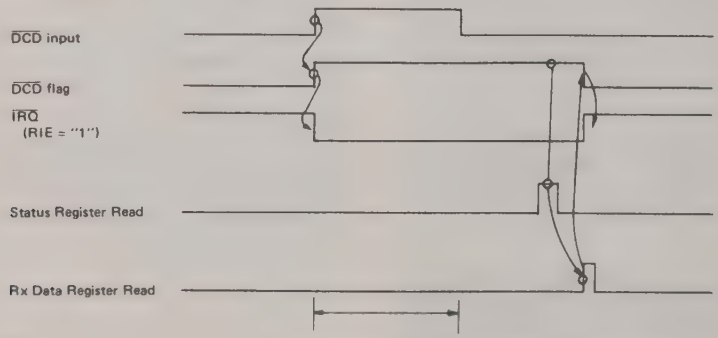

DCD input
DCD flag
IRQ
(RIE = "1")
Status Register Read
Rx Data Register Read
All the receiving operation are prohibited and ACIA is stopped in this period.

Figure 16 DCD Flag Timing Chart

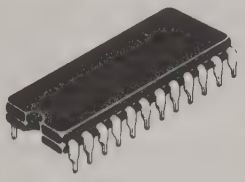
HD6350,HD63A50,HD63B50
CMOS ACIA (Asynchronous Communications Interface Adapter)

The HD6350 CMOS Asynchronous Communications Interface Adapter provides the data formatting and control to interface serial asynchronous data communications information to bus organized systems such as the HMCS6800 Microprocessing Unit.

The bus interface of the HD6350 includes select, enable, read/write, interrupt and bus interface logic to allow data transfer over an 8-bit bi-directional data bus. The parallel data of the bus system is serially transmitted and received by the asynchronous data interface, with proper formatting and error checking.

The functional configuration of the ACIA is programmed via the data bus during system initialization. A programmable Control Register provides variable word lengths, clock division ratios, transmit control, receive control, and interrupt control. For peripheral or modem operation three control lines are provided. Exceeding Low Power dissipation is realized due to adopting CMOS process.

HD6350P, HD63A50P, HD63B50P

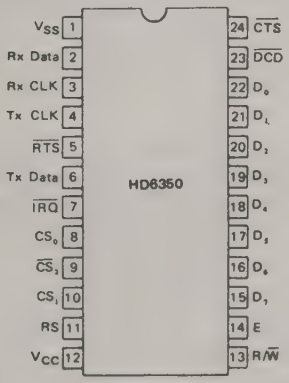


(DP-24)

■ FEATURES

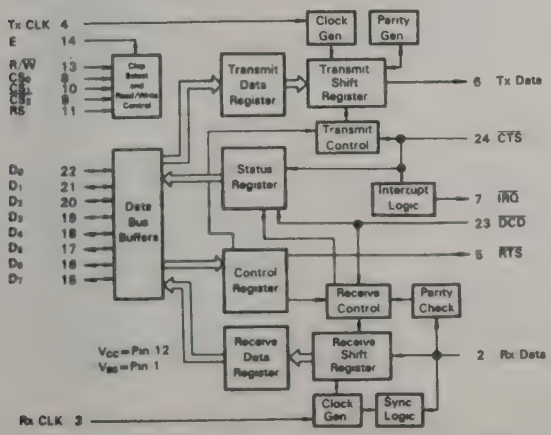
- Low-Power, High-Speed, High-Density CMOS
- Compatible with NMOS ACIA (HD6850)
- Serial/Parallel Conversion of Data
- Eight and Nine-bit Transmission
- Optional Even and Odd Parity
- Parity, Overrun and Framing Error Checking
- Peripheral/Modem Control Functions (Clear to Send $\overline{CTS}$, Request to Send $\overline{RTS}$, Data Carrier Detect $\overline{DCD}$)
- Optional ÷ 1, ÷ 16, and ÷ 64 Clock Modes
- Up to 500kbps Transmission

■ PIN ARRANGEMENT



(Top View)

■ BLOCK DIAGRAM



■ ABSOLUTE MAXIMUM RATINGS

Item	Symbol	Value	Unit
Supply Voltage	V_{CC}*	-0.3 ~ +7.0	V
Input Voltage	V_{in}*	-0.3 ~ +7.0	V
Maximum Output Current**	$\|I_O\|$	10	mA
Operating Temperature	T_{opr}	-20 ~ +75	°C
Storage Temperature	T_{stg}	-55 ~ +150	°C

* With respect to V_{SS} (SYSTEM GND)
** Maximum output current is the maximum current which can flow out from one output terminal or I/O common terminal ($D_0 \sim D_7$, $\overline{RTS}$, Tx Data, $\overline{IRQ}$).

(NOTE) Permanent LSI damage may occur if maximum ratings are exceeded. Normal operation should be under recommended operating conditions. If these conditions are exceeded, it could affect reliability of LSI.

■ RECOMMENDED OPERATING CONDITIONS

Item		Symbol	min	typ	max	Unit
Supply Voltage		V_{CC}*	4.5	5.0	5.5	V
Input "Low" Voltage		V_{IL}*	0	–	0.8	V
Input "High" Voltage	$D_0 \sim D_7$, RS, Tx CLK, $\overline{DCD}$, $\overline{CTS}$, Rx Data	V_{IH}*	2.0	–	V_{CC}	V
	CS_0, $\overline{CS_2}$, CS_1, R/$\overline{W}$, E, Rx CLK		2.2	–	V_{CC}	
Operating Temperature		T_{opr}	-20	25	75	°C

* With respect to V_{SS} (SYSTEM GND)

■ ELECTRICAL CHARACTERISTICS

● DC CHARACTERISTICS (V_{CC} = 5V±10%, V_{SS} = 0V, T_a = -20 ~ +75°C, unless otherwise noted.)

Item		Symbol	Test Condition	min	typ	max	Unit
Input "High" Voltage	$D_0 \sim D_7$, RS, Tx CLK, $\overline{DCD}$, $\overline{CTS}$, Rx Data	V_{IH}		2.0	–	V_{CC}	V
	CS_0, $\overline{CS_2}$, CS_1, R/$\overline{W}$, E, Rx CLK			2.2	–	V_{CC}	
Input "Low" Voltage	All Inputs	V_{IL}		-0.3	–	0.8	V
Input Leakage Current	R/$\overline{W}$, CS_0, CS_1, $\overline{CS_2}$, E	I_{in}	V_{in} = 0 ~ V_{CC}	-2.5	–	2.5	μA
Three-State (Off State) Input Current	$D_0 \sim D_7$	I_{TSI}	V_{in} = 0.4 ~ V_{CC}	-10	–	10	μA
Output "High" Voltage	$D_0 \sim D_7$	V_{OH}	I_{OH} = -400 μA	4.1	–	–	V
			$I_{OH} \leq$ -10 μA	V_{CC}-0.1	–	–	
	Tx Data, $\overline{RTS}$		I_{OH} = -400 μA	4.1	–	–	
			$I_{OH} \leq$ -10 μA	V_{CC}-0.1	–	–	
Output "Low" Voltage	All Outputs	V_{OL}	I_{OH} = 1.6 mA	–	–	0.4	V
Output Leakage Current (Off State)	$\overline{IRQ}$	I_{LOH}	V_{OH} = V_{CC}	–	–	10	μA
Input Capacitance	$D_0 \sim D_7$	C_{in}	V_{in} = 0V, T_a = 25°C, f = 1.0 MHz	–	–	12.5	pF
	E, Tx CLK, Rx CLK, R/$\overline{W}$, RS, Rx Data, CS_0, CS_1, $\overline{CS_2}$, $\overline{CTS}$, $\overline{DCD}$			–	–	7.5	
Output Capacitance	$\overline{RTS}$, Tx Data	C_{out}	V_{in} = 0V, T_a = 25°C, f = 1.0 MHz	–	–	10	pF
	$\overline{IRQ}$			–	–	50	
Supply Current	• Under transmitting and Receiving operation • 500 kbps • Data bus in R/$\overline{W}$ operation	I_{CC}	E = 1.0 MHz	–	–	3	mA
			E = 1.5 MHz	–	–	4	
			E = 2.0 MHz	–	–	5	
	• Chip is not selected • 500 kbps • Under non transmitting and receiving operation • Input level (Except E) V_{IH} min = V_{CC}-0.8V V_{IL} max = 0.8V		E = 1.0 MHz	–	–	200	μA
			E = 1.5 MHz	–	–	250	
			E = 2.0 MHz	–	–	300	

- **AC CHARACTERISTICS (V_{CC} = 5.0V ±10%, V_{SS} = 0V, T_a = -20 ~ +75°C, unless otherwise noted.)**

1. TIMING OF DATA TRANSMISSION

Item		Symbol	Test Condition	HD6350		HD63A50		HD63B50		Unit
				min	max	min	max	min	max	
Minimum Clock Pulse Width	÷ 1 Mode	PW_{CL}	Fig. 1	900	–	650	–	500	–	ns
	÷ 16, ÷ 64 Modes			600	–	450	–	280	–	ns
	÷ 1 Mode	PW_{CH}	Fig. 2	900	–	650	–	500	–	ns
	÷ 16, ÷ 64 Modes			600	–	450	–	280	–	ns
Clock Frequency	÷ 1 Mode	f_C		–	500	–	750	–	1000	kHz
	÷ 16, ÷ 64 Modes			–	800	–	1000	–	1500	kHz
Clock-to-Data Delay for Transmitter		t_{TDD}	Fig. 3	–	600	–	540	–	460	ns
Receive Data Setup Time	÷ 1 Mode	t_{RDSU}	Fig. 4	250	–	100	–	30	–	ns
Receive Data Hold Time	÷ 1 Mode	t_{RDH}	Fig. 5	250	–	100	–	30	–	ns
$\overline{IRQ}$ Release Time		t_{IR}	Fig. 6	–	1200	–	900	–	700	ns
RTS Delay Time		t_{RTS}	Fig. 6	–	560	–	480	–	400	ns
Rise Time and Fall Time	Except E	t_r, t_f		–	1000*	–	500	–	250	ns

* 1.0 μs or 10% of the pulse width, whichever is smaller.

2. BUS TIMING CHARACTERISTICS

1) READ

Item	Symbol	Test Condition	HD6350		HD63A50		HD63B50		Unit
			min	max	min	max	min	max	
Enable Cycle Time	t_{cycE}	Fig. 7	1000	–	666	–	500	–	ns
Enable "High" Pulse Width	PW_{EH}	Fig. 7	450	–	280	–	220	–	ns
Enable "Low" Pulse Width	PW_{EL}	Fig. 7	430	–	280	–	210	–	ns
Setup Time, Address and R/$\overline{W}$ Valid to Enable Positive Transition	t_{AS}	Fig. 7	80	–	60	–	40	–	ns
Data Delay Time	t_{DDR}	Fig. 7	–	290	–	180	–	150	ns
Data Hold Time	t_H	Fig. 7	20	100	20	100	20	100	ns
Address Hold Time	t_{AH}	Fig. 7	10	–	10	–	10	–	ns
Rise and Fall Time for Enable Input	t_{Er}, t_{Ef}	Fig. 7	–	25	–	25	–	20	ns

2) WRITE

Item	Symbol	Test Condition	HD6350		HD63A50		HD63B50		Unit
			min	max	min	max	min	max	
Enable Cycle Time	t_{cycE}	Fig. 8	1000	–	666	–	500	–	ns
Enable "High" Pulse Width	PW_{EH}	Fig. 8	450	–	280	–	220	–	ns
Enable "Low" Pulse Width	PW_{EL}	Fig. 8	430	–	280	–	210	–	ns
Setup Time, Address and R/$\overline{W}$ Valid to Enable Positive Transition	t_{AS}	Fig. 8	80	–	60	–	40	–	ns
Data Setup Time	t_{DSW}	Fig. 8	165	–	80	–	60	–	ns
Data Hold Time	t_H	Fig. 8	10	–	10	–	10	–	ns
Address Hold Time	t_{AH}	Fig. 8	10	–	10	–	10	–	ns
Rise and Fall Time for Enable Input	t_{Er}, t_{Ef}	Fig. 8	–	25	–	25	–	20	ns

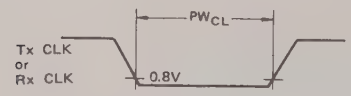


Figure 1 Clock Pulse Width, "Low" State

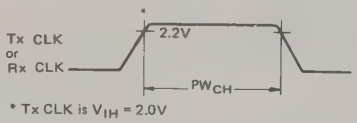


Figure 2 Clock Pulse Width, "High" State

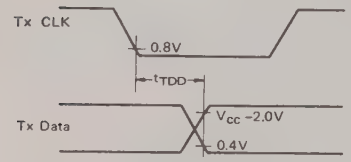


Figure 3 Transmit Data Output Delay

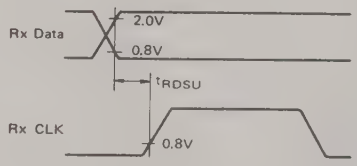


Figure 4 Receive Data Setup Time (÷1 Mode)

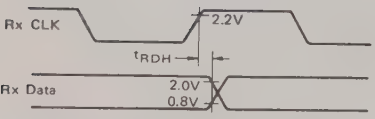
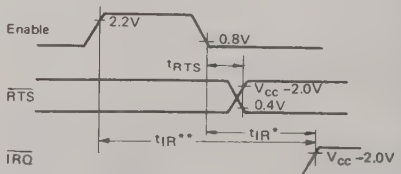


Figure 5 Receive Data Hold Time (÷1 Mode)



* (1) $\overline{IRQ}$ Release Time applied to Rx Data Register read operation.
(2) $\overline{IRQ}$ Release Time applied to Tx Data Register write operation.
(3) $\overline{IRQ}$ Release Time applied to control Register write TIE = 0, RIE = 0 operation.

** $\overline{IRQ}$ Release Time applied to Rx Data Register read operation right after read status register, when $\overline{IRQ}$ is asserted by $\overline{DCD}$ rising edge.

(Note) Note that followings take place when $\overline{IRQ}$ is asserted by the detection of transmit data register empty status. $\overline{IRQ}$ is released to "High" asynchronously with E signal when $\overline{CTS}$ goes "High". (Refer to Figure 14)

Figure 6 $\overline{RTS}$ Delay and $\overline{IRQ}$ Release Time

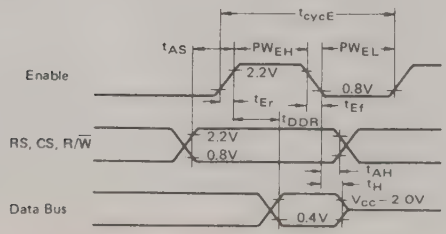


Figure 7 Bus Read Timing Characteristics
(Read information from ACIA)

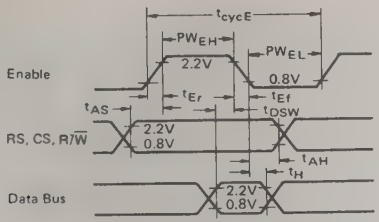
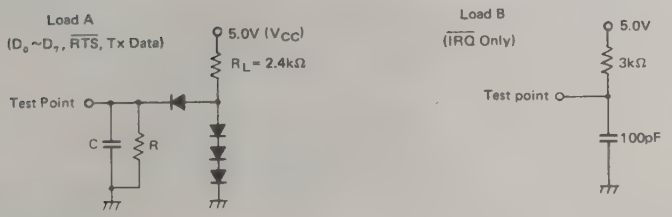


Figure 8 Bus Write Timing Characteristics
(Write information into ACIA)



C = 130pF for D_0~D_7
= 30pF for $\overline{RTS}$ and Tx Data
R = 10kΩ for D_0 ~ D_7, $\overline{RTS}$ and Tx Data
All diodes are 1S2074Ⓗ or Equivalent.

Figure 9 Bus Timing Test Loads

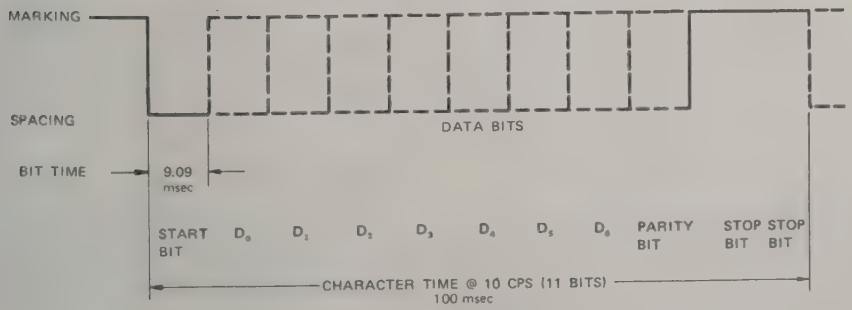
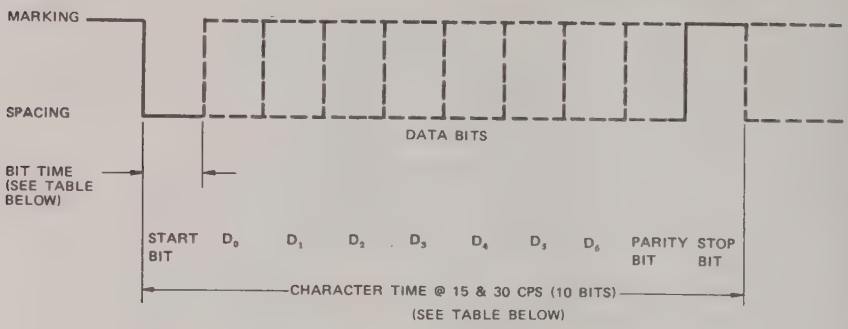


Figure 10 110 Baud Serial ASCII Data Timing



BAUD RATE	150	300
CHARACTERS/SEC	15	30
BIT TIME (msec)	6.67	3.33
CHARACTER TIME (msec)	66.7	33.3

$$\text{BIT TIME} = \frac{\text{SEC}}{\text{BAUD RATE}}$$

Figure 11 150 & 300 Baud Serial ASCII Data Timing

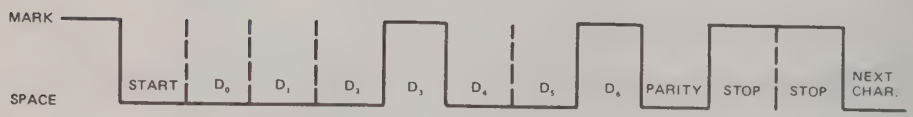


Figure 12 Send a 7 Bit ASCII Char. "H" Even Parity
– 2 Stop Bits H = 48_{16} = 1001000_2

■ DATA OF ACIA

HD6350 is an interface adapter which controls transmission and reception of Asynchronous serial data. Some examples of serial data are shown in Figs. 10 ~ 12.

■ INTERNAL STRUCTURE OF ACIA

HD6350(ACIA) provides the following; 8-bit Bi-directional Data Buses (D_0~D_7), Receive Data Input (Rx Data), Transmit Data Output (Tx Data), three Chip Selects (CS_0, CS_1, $\overline{CS_2}$), Register Select Input (RS), Two Control Input (Read/Write: R/$\overline{W}$, Enable: E), Interrupt Request Output($\overline{IRQ}$), Clear-to-Send ($\overline{CTS}$) to control the modem, Request-to-Send ($\overline{RTS}$), Data Carrier Detect($\overline{DCD}$) and Clock Inputs(Tx CLK, Rx CLK) used for synchronization of received and transmitted data. This ACIA also provides four registers; Status Register, Control Register, Receive Register and Transmit Register.

24-pin dual-in-line type package is used for the ACIA. Internal Structure of ACIA is illustrated in Fig. 13.

■ ACIA OPERATION

• Master Reset

ACIA has an internal master reset function controlled by software, since it has no hardware reset pin. Bit 0 and bit 1 of control register should be set to "11" to execute master reset, also bit 5 and bit 6 should be programmed to get predetermined $\overline{RTS}$ output accordingly. To release the master reset, the data other than "11" should be written into bit 0, bit 1 of the control register. When the master reset is released, the control register needs to be programmed to get predetermined options such as clock divider ratios, word length, one or two stop bits, parity(even, old, or none), etc.

It may happen that "Low" level output is provided in $\overline{IRQ}$ pin during the time after power-on till master reset. In the system using ACIA, interrupt mask bit of MPU should be released after the master reset of ACIA. (MPU interrupt should be prohibited until MPU program completes the master reset of ACIA.) Transmit Data Register (TDR) and Receive Data Register (RDR) can not be reset by master reset.

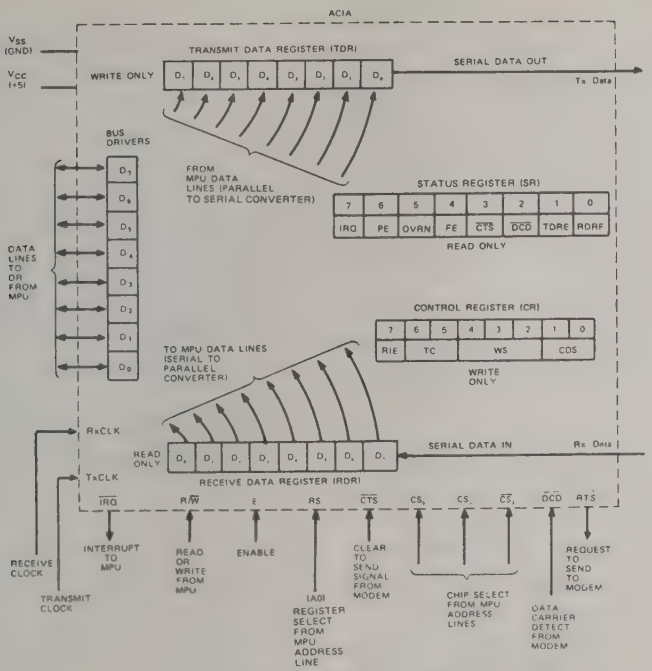


Figure 13 Internal Structure of ACIA

- **Transmit**

A typical transmitting sequence consists of reading the ACIA Status Register either as a result of an interrupt or in the ACIA's turn in a polling sequence. A character may be written into the Transmit Data Register if the status read operation has indicated that the Transmit Data Register is empty. This character is transferred to a Shift Register where it is serialized and transmitted from the Transmit Data output preceded by a start bit and followed by one or two stop bits. Internal parity (odd or even) can be optionally added to the character and will occur between the last data bit and the first stop bit. After the first character is written in the Data Register, the Status Register can be read again to check for a Transmit Data Register Empty condition and current peripheral status. If the register is empty, another character can be loaded for transmission even though the first character is in the process of being transmitted (because of double buffering). The second character will be automatically transferred into the Shift Register when the first character transmission is completed. This sequence continues until all the characters have been transmitted.

- **Receive**

Data is received from a peripheral by means of the Receive Data input. A divide-by-one clock ratio is provided for an externally synchronized clock (to its data) while the divide-by-16 and 64 ratios are provided for internal synchronization. Bit synchronization in the divide-by-16 and 64 modes is initiated by the detection of the leading mark-space transition of the start bit. False start bit delection capability insures that a full half bit of a start bit has been received before the internal clock is synchronized to the bit time. As a character is being received, parity (odd or even) will be checked and the error indication will be available in the Status Register along with framing error, overrun error, and Receive Data Register full. In a typical receiving sequence, the Status Register is read to determine if a character has been received from a peripheral. If the Receiver Data Register is full, the character is placed on the 8-bit ACIA bus when a Read Data command is received from the MPU. When parity has been selected for an 8-bit word (7 bits plus parity), the receiver strip the parity bit (D_7="0") so that data alone is transferred to the MPU. This feature reduces MPU programming. The Status Register can continue to be read again to determine when another character is available in the Receive Data Register. The receiver is also double buffered so that a character can be read from the data register as another character is being received in the Shift register. The above sequence continues until all characters have been received.

■ ACIA INTERNAL REGISTERS

The ACIA provides four registers; Transmit Data Register (TDR), Receive Data Register(RDR), Control Register(CR) and Status Register(SR). The content of each of the registers is summarized in Table 1.

Table 1 Definition of ACIA Register Contents

Buffer Address	RS=1 · R/$\overline{W}$=0 ****	RS=1 · R/$\overline{W}$=1	RS=0 · R/$\overline{W}$=0	RS=0 · R/$\overline{W}$=1
Data Bus	Transmit Data Register	Receiver Data Register	Control Register	Status Register
	(Write Only)	(Read Only)	(Write Only)	(Read Only)
0	Data Bit 0*	Data Bit 0	Counter Divide Select (CR0)	Rx Data Reg. Full (RDRF)
1	Data Bit 1	Data Bit 1	Counter Divide Select (CR1)	Tx Data Reg. Empty (TDRE)
2	Data Bit 2	Data Bit 2	Word Select 1 (CR2)	Data Carrier Detect ($\overline{DCD}$)
3	Data Bit 3	Data Bit 3	Word Select 2 (CR3)	Clear to Send ($\overline{CTS}$)
4	Data Bit 4	Data Bit 4	Word Select 3 (CR4)	Framing Error (FE)
5	Data Bit 5	Data Bit 5	Tx Control 1 (CR5)	Overrun (OVRN)
6	Data Bit 6	Data Bit 6	Tx Control 2 (CR6)	Parity Error (PE)
7	Data Bit 7***	Data Bit 7**	Rx Interrupt Enable (CR7)	Interrupt Request ($\overline{IRQ}$)

* Leading bit = LSB = Bit 0
** Data bit will be zero in 7-bit plus parity modes.
*** Data bit is "don't care" in 7-bit plus parity modes.
**** 1 ... "High" level, 0 ... "Low" level

- **Transmit Data Register (TDR)**

Data is written in the Transmit Data Register during the negative transition of the enable (E) when the ACIA has been addressed and RS · R/$\overline{W}$ is selected. Writing data into the register causes the Transmit Data Register Empty bit in the Status Register to go "0". Data can then be transmitted. If the transmitter is idling and no character is being transmitted, then the transfer will take place within 2 bit time + several E cycles of the trailing edge of the Write command. If a character is being transmitted, the new data character will commence as soon as the previous character is complete. The transfer of data causes the Transmit Data Register Empty (TDRE) bit to indicate empty.

- **Receive Data Register (RDR)**

Data is automatically transferred to the empty Receive Data Register (RDR) from the receiver deserializer (a shift register) upon receiving a complete character. This event causes the Receive Data Register Full bit (RDRF) on the status buffer to go "1" (full). Data may then be read through the bus by addressing the ACIA and R/$\overline{W}$ "High" when the ACIA is enabled. The non-destructive read cycle causes the RDRF bit to be cleared to empty although the data is retained in the RDR. The status is maintained by RDRF as to whether or not the data is current. When the Receive Data Register is full, the automatic transfer of data from the Receiver Shift Register to the Data Register is inhibited and the RDR contents remain valid with its current status stored in the Status Register.

- **Control Register**

The ACIA Control Register consists of eight bits of write-only buffer that are selected when RS and R/$\overline{W}$ are "Low". This register controls the function of the receiver, transmitter, interrupt enables, and the Request-to-Send ($\overline{RTS}$) peripheral/modem control output.

Counter Divide Select Bits (CR0 and CR1)

The Counter Divide Select Bits (CR0 and CR1) determine the divide ratios utilized in both the transmitter and receiver section of the ACIA. Additionally, these bits are used to provide a master reset for the ACIA which clears the Status Register (except for external conditions on $\overline{CTS}$ and $\overline{DCD}$) and initializes both the receiver and transmitter. Master reset does not affect other Control Register bits. Note that after power-on or a power fail/restart, these bits must be set "1" to reset the ACIA. After resetting, the clock divide ratio may be selected. These counter select bits provide for the following clock divide ratios:

Table 2 Function of Counter Devide Select Bit

CR1	CR0	Function
0	0	÷1
0	1	÷16
1	0	÷64
1	1	Master Reset

Word Select Bits (CR2, CR3, and CR4)

The Word Select bits are used to select word length, parity, and the number of stop bits. The encoding format is as follows:

Table 3 Function of Word Select Bit

CR4	CR3	CR2	Function
0	0	0	7 Bits + Even Parity + 2 Stop Bits
0	0	1	7 Bits + Odd Parity + 2 Stop Bits
0	1	0	7 Bits + Even Parity + 1 Stop Bit
0	1	1	7 Bits + Odd Parity + 1 Stop Bit
1	0	0	8 Bits + 2 Stop Bits
1	0	1	8 Bits + 1 Stop Bit
1	1	0	8 Bits + Even Parity + 1 Stop Bit
1	1	1	8 Bits + Odd Parity + 1 Stop Bit

Word length, Parity Select, and Stop Bit changes are not buffered and therefore become effective immediately.

Transmitter Control Bits (CR5 and CR6)

Two Transmitter Control bits provide for the control of the interrupt from the Transmit Data Register Empty condition, the Request-to-Send ($\overline{RTS}$) output, and the transmission of a Break level (space). The following encoding format is used:

Table 4 Function of Transmitter Control-Bit

CR6	CR5	Function
0	0	$\overline{RTS}$ = "Low", Transmitting Interrupt Disabled.
0	1	$\overline{RTS}$ = "Low", Transmitting Interrupt Enabled.
1	0	$\overline{RTS}$ = "High", Transmitting Interrupt Disabled.
1	1	$\overline{RTS}$ = "Low", Transmits a Break level on the Transmit Data Output. Transmitting Interrupt Disabled.

Receive Interrupt Enable Bit (CR7)

The following interrupts will be enabled by a "1" in bit position 7 of the Control Register (CR7): Receive Data Register Full, Overrun, or a "Low" to "High" transistion on the Data Carrier Detect ($\overline{DCD}$) signal line.

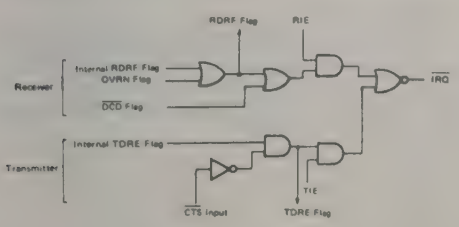


Figure 14 $\overline{IRQ}$ Internal Circuit

- **Status Register**

Information on the status of the ACIA is available to the MPU by reading the ACIA Status Register. This read-only register is selected when RS is "Low" and R/$\overline{W}$ is "High". Information stored in this register indicates the status of the Transmit Data Register, the Receive Data Register and error logic, and the peripheral/modem status inputs of the ACIA.

Receive Data Register Full (RDRF), Bit 0

RDRF indicates that received data has been transferred to the Receive Data Register. RDRF is cleared after an MPU read of the Receive Data Register or by a master reset. The cleared or empty state indicates that the contents of the Receive Data Register are not current. Data Carrier Detect ($\overline{DCD}$) being "High" also causes RDRF to indicate empty.

Transmit Data Register Empty (TDRE), Bit 1

The Transmit Data Register Empty bit being set "1" indicates that the Transmit Data Register contents have been transferred and that new data may be entered. The "0" state indicates that the register is full and that transmission of a new character has not begun since the last write data command.

Data Carrier Detect ($\overline{DCD}$), Bit 2

The $\overline{DCD}$ bit will be "1" when the $\overline{DCD}$ input from a modem has gone "High" to indicate that a carrier is not present. This bit going "1" causes an Interrupt Request to be generated when the Receive Interrupt Enable is set. It remains "1" after the $\overline{DCD}$ input is returned "Low" until cleared by first reading the Status Register and then the Data Register or until a master reset occurs. If the $\overline{DCD}$ input remains "High" after read status and read data or master reset has occurred, the interrupt is cleared, the $\overline{DCD}$ status bit remains "1" and will follow the $\overline{DCD}$ input.

Clear-to-Send ($\overline{CTS}$), Bit 3

The $\overline{CTS}$ bit indicates the state of the $\overline{CTS}$ input from a modem. A "Low" $\overline{CTS}$ input indicates that there is a $\overline{CTS}$ from the modem. In the "High" state, the Transmit Data Register Empty bit is inhibited and the $\overline{CTS}$ status bit will be "1". Master reset does not affect the Clear-to-Send Status bit.

Framing Error (FE), Bit 4

FE flag indicates that the received character is improperly framed by a start and a stop bit and is detected by the absence of the 1st stop bit. This error indicates a synchronization error, faulty transmission, or a break condition. The FE flag is set or reset during the receive data transfer time. Therefore, this error indicator is present throughout the time that the associated character is available.

Receiver Overrun (OVRN), Bit 5

Overrun is an error flag that indicates that one or more characters in the data stream were lost. That is, a character or a number of characters were received but not read from the Receive Data Register (RDR) prior to subsequent characters being received. The overrun condition begins at the midpoint of the last bit of the second character received in succession without a read of the RDR having occurred. The overrun does not occur in the Status Register until the valid character prior to Overrun has been read. The RDRF bit remains set until the Overrun is reset. Character synchronization is maintained during the Overrun condition. The Overrun indication is reset after the reading of data from the Receive Data Register or by a Master Reset.

Parity Error (PE), Bit 6

The PE flag indicates that the number of "1"s (highs) in the character does not agree with the preselected odd or even parity. Odd parity is defined to be when the total number of ones is odd. The parity error indication will be present as long as the data character is in the RDR. If no parity is selected, then both the transmitter parity generator output and the receiver parity check results are inhibited.

Interrupt Request (IRQ), Bit 7

The IRQ bit indicates the state of the $\overline{IRQ}$ output. Any interrupt condition with its applicable enable will be indicated in this status bit. Anytime the $\overline{IRQ}$ output is "Low" the IRQ bit will be "1" to indicate the interrupt or service request status. IRQ is cleared by a read operation to the Receive Data Register or a write operation to the Transmit Data Register. (Refer to Figure 14.)

■ SIGNAL FUNCTIONS

• Interface Signal for MPU

Bi-Directional Data Bus (D_0~D_7)

The bi-directional data bus (D_0~D_7) allow for data transfer between the ACIA and the MPU. The data bus output drivers are three-state devices that remain in the high impedance (off) state except when the MPU performs an ACIA read operation.

Enable (E)

The Enable signal, E, is a high impedance TTL compatible input that enables the bus input/output data buffers and clocks data to and from the ACIA. This signal will normally be a derivative of the HMCS6800 ϕ_2 Clock. The ACIA accepts both continuous pulse signal and strobe type signal as Enable input.

Read/Write (R/W)

The R/$\overline{W}$ line is a high impedance input that is TTL compatible and is used to control the direction of data flow through the ACIA's input/output data bus interface. When R/$\overline{W}$ is "High" (MPU Read cycle), ACIA output drivers are turned on and a selected register is read. When it is "Low", the ACIA output drivers are turned off and the MPU writes into a selected register. Therefore, the R/$\overline{W}$ signal is used to select read-only or write-only registers within the ACIA.

Chip Select (CS_0, CS_1, $\overline{CS_2}$)

These three high impedance TTL compatible input lines are used to address the ACIA. The ACIA is selected when CS_0 and CS_1 are "High" and $\overline{CS_2}$ is "Low". Transfers of data to and from the ACIA are then performed under the control of the Enable signal, Read/Write, and Register Select.

Register Select (RS)

The RS line is a high impedance input that is TTL compatible. A "High" level is used to select the Transmit/Receive Data Registers and a "Low" level the Control/Status Registers. The R/$\overline{W}$ signal line is used in conjunction with Register Select to select the read-only or write-only register in each register pair.

Interrupt Request ($\overline{IRQ}$)

$\overline{IRQ}$ is a TTL compatible, open-drain (no internal pullup), active "Low" output that is used to interrupt the MPU. The $\overline{IRQ}$ output remains "Low" as long as the cause of the interrupt is present and the appropriate interrupt enable within the ACIA is set.

Clock Inputs

Separate high impedance TTL compatible inputs are provided for clocking of transmitted and received data. Clock frequencies of 1, 16 or 64 times the data rate may be selected.

Transmit Clock (Tx CLK)

The Tx CLK input is used for the clocking of transmitted data. The transmitter initiates data on the negative transition of the clock.

Receive Clock (Rx CLK)

The Rx CLK input is used for synchronization of received data. (In the ÷ 1 mode, the clock and data must be synchronized externally.) The receiver samples the data on the positive transition of the clock.

• Serial Input/Output Lines

Receive Data (Rx Data)

The Rx Data line is a high impedance TTL compatible input through which data is received in a serial format. Synchronization with a clock for detection of data is accomplished internally when clock rates of 16 or 64 times the bit rate are used. Data rates are in the range of 0 to 500 kbps when external synchronization is utilized.

Transmit Data (Tx Data)

The Tx Data output line transfers serial data to a modem or other peripheral. Data rates in the range of 0 to 500 kbps when external synchronization is utilized.

Modem Control

The ACIA includes several functions that permit limited control of a peripheral or modem. The functions included are $\overline{CTS}$, $\overline{RTS}$ and $\overline{DCD}$.

Clear-to-Send ($\overline{CTS}$)

This high impedance TTL compatible input provides automatic control of the transmitting end of a communications link via the modem $\overline{CTS}$ active "Low" output by inhibiting the Transmit Data Register Empty (TDRE) status bit. (Refer to Figure 15.)

Request-to-Send ($\overline{RTS}$)

The $\overline{RTS}$ output enables the MPU to control a peripheral or modem via the data bus. The $\overline{RTS}$ output corresponds to the state of the Control Register bits CR5 and CR6. When CR6=0 or both CR5 and CR6=1, the $\overline{RTS}$ output is "Low" (the active state). This output can also be used for Data Terminal Ready ($\overline{DTR}$). (Refer to Figure 15.)

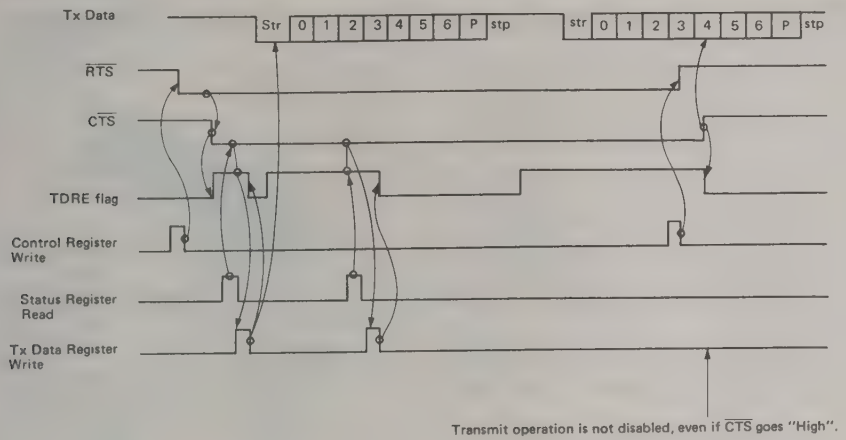

Tx Data
Str 0 1 2 3 4 5 6 P stp
str 0 1 2 3 4 5 6 P stp
RTS
CTS
TDRE flag
Control Register Write
Status Register Read
Tx Data Register Write
Transmit operation is not disabled, even if CTS goes "High".

Figure 15 RTS and CTS Timing Chart (Example of 2 bytes transmission)

Data Carrier Detect (DCD)

DCD is the input signal corresponding to the "carrier detect" signal which shows carrier detect of modem.

DCD signal is used to control the receiving operation. When DCD input goes "High", ACIA stops all the receiving operation and sets receiving part in reset status. It means that receive shift register stops shifting, error detection circuit and synchro. zation circuit of receive clock are reset. When DCD is in "High" level, the receiving part of ACIA is kept in initial status and the operation in the receiving part is prohibited. When DCD goes "Low", the receiving part is allowed to receive data. In this case, the following process is needed to reset DCD flag and restarts the receive operation. (Refer to Figure 16.)

(1) Return DCD input from "High" to "Low".
(2) Read status register. (DCD flag = "1")
(3) Read receive data register (Uncertain data will be read.)

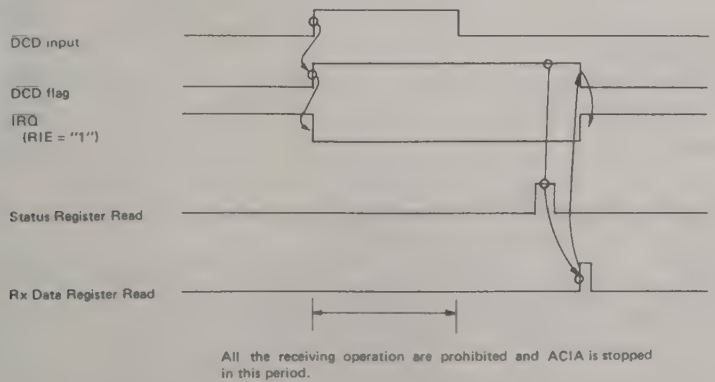

DCD input
DCD flag
IRQ (RIE = "1")
Status Register Read
Rx Data Register Read
All the receiving operation are prohibited and ACIA is stopped in this period.

Figure 16 DCD Flag Timing Chart

■ **Note for Use**

Input Signal, which is not necessary for user's application, should be used fixed to "High" or "Low" level. This is applicable to the following signal pins.
Rx Data, Rx CLK, Tx CLK, CTS, DCD

HD6852, HD68A52

SSDA (Synchronous Serial Data Adapter)

The HD6852 Synchronous Serial Data Adapter provides a bi-directional serial interface for synchronous data information interchange. It contains interface logic for simultaneously transmitting and receiving standard synchronous communications characters in bus organized systems such as the HMCS6800 Microprocessor systems.

The bus interface of the HD6852 includes select, enable, read/write, interrupt, and bus interface logic to allow data transfer over an 8-bit bi-directional data bus. The parallel data of the bus system is serially transmitted and received by the synchronous data interface with synchronization, fill character insertion/deletion, and error checking. The functional configuration of the SSDA is programmed via the data bus during system initialization.

Programmable control registers provide control for variable word length, transmit control, receive control, synchronization control and interrupt control. Status, timing and control lines provide peripheral or modem control.

Typical applications include data communications terminals, floppy disk controllers, cassette or cartridge tape controllers and numerical control systems.

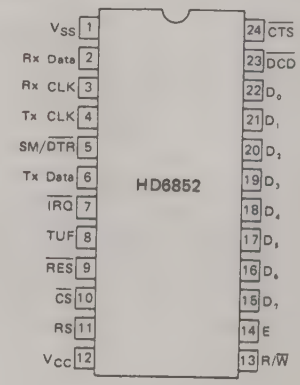
HD6852P, HD68A52P

(DP-24)

■ FEATURES

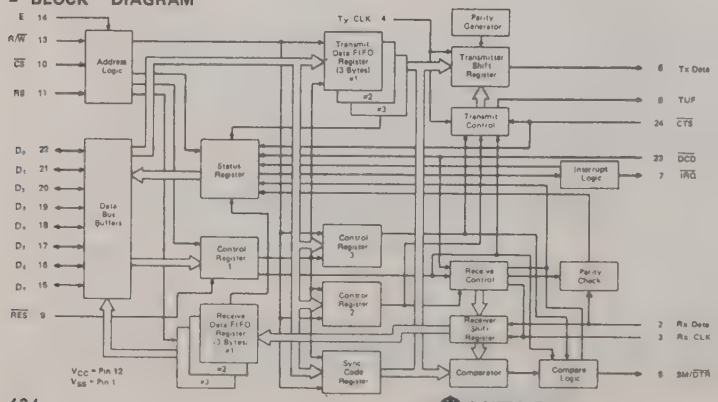
- Programmable Interrupts from Transmitter, Receiver, and Error Detection Logic
- Character Synchronization on One or Two Sync Codes
- External Synchronization Available for Parallel-Serial Operation
- Programmable Sync Code Register
- Up to 600kbps Transmitter
- Peripheral/Modem Control Functions
- Three Bytes of FIFO Buffering on Both Transmit and Receive
- 6, 7, or 8 Bit Data Transmission
- Optional Even and Odd Parity
- Parity, Overrun, and Underflow Status
- Compatible with MC6852 and MC68A52

■ PIN ARRANGEMENT



(Top View)

■ BLOCK DIAGRAM



■ ABSOLUTE MAXIMUM RATINGS

Item	Symbol	Value	Unit
Supply Voltage	V_{CC}*	−0.3 ~ +7.0	V
Input Voltage	V_{in}*	−0.3 ~ +7.0	V
Operating Temperature	T_{opr}	− 20 ~ + 75	°C
Storage Temperature	T_{stg}	−55 ~ +150	°C

* With respect to V_{SS} (SYSTEM GND)

(NOTE) Permanent LSI damage may occur if maximum ratings are exceeded. Normal operation should be under recommended operating conditions. If these conditions are exceeded, it could affect reliability of LSI.

■ RECOMMENDED OPERATING CONDITIONS

Item	Symbol	min	typ	max	Unit
Supply Voltage	V_{CC}*	4.75	5.0	5.25	V
Input Voltage	V_{IL}*	−0.3	–	0.8	V
	V_{IH}*	2.0	–	V_{CC}	V
Operating Temperature	T_{opr}	− 20	25	75	°C

* With respect to V_{SS} (SYSTEM GND)

■ ELECTRICAL CHARACTERISTICS

● DC CHARACTERISTICS (V_{CC} = 5V ± 5%, V_{SS} = 0V, Ta = −20~+75°C, unless otherwise noted.)

Item		Symbol	Test Condition	min	typ*	max	Unit
Input "High" Voltage	All Input	V_{IH}		2.0	–	–	V
Input "Low" Voltage	All Input	V_{IL}		−0.3	–	0.8	V
Output "High" Voltage	D_0~D_7	V_{OH}	I_{OH} = −205 μA, PW_{EH}, $PW_{EL} \leqq 25\mu s$	2.4	–	–	V
	Tx Data $\overline{DTR}$, TUF	V_{OH}	I_{OH} = −100 μA, PW_{EH}, $PW_{EL} \leqq 25\mu s$	2.4	–	–	V
Output "Low" Voltage	All Output	V_{OL}	I_{OL} = 1.6 mA, PW_{EH}, $PW_{EL} \leqq 25\mu s$	–	–	0.4	V
Input Leakage Current	TxCLK, RxCLK, Rx Data, E, $\overline{RES}$, RS, R/$\overline{W}$ $\overline{CS}$, $\overline{DCD}$, $\overline{CTS}$	I_{in}	V_{in} = 0 ~ 5.25V	-2.5	–	2.5	μA
Three-State Input Current (Off State)	D_0~D_7	I_{TSI}	V_{in} = 0.4 ~ 2.4V, V_{CC} = 5.25V	-10	–	10	μA
Output Leakage Current (Off State)	$\overline{IRQ}$	I_{LOH}	V_{OH} = 2.4V	–	–	10	μA
Power Dissipation		P_D		–	300	525	mW
Input Capacitance	D_0~D_7	C_{in}	V_{in} = 0V, Ta = 25°C, f = 1 MHz	–	–	12.5	pF
	RxData, RxCLK, TxCLK, $\overline{RES}$, $\overline{CS}$, RS, R/$\overline{W}$, E, $\overline{DCD}$, $\overline{CTS}$			–	–	7.5	
Output Capacitance	TxData, $\overline{DTR}$, TUF, $\overline{IRQ}$	C_{out}	V_{in} = 0V, Ta = 25°C f = 1 MHz	–	–	10	pF
				–	–	5.0	

* Ta = 25°C, V_{CC} = 5V

- **AC CHARACTERISTICS** (V_{CC}=5V±5%, V_{SS}=0V, Ta=−20~+75°C, unless otherwise noted.)

1. TIMING OF THE DATA TRANSFER

Item	Symbol	Test Condition	HD6852			HD68A52			Unit
			min	typ	max	min	typ	max	
Clock "Low" Pulse Width	PW_{CL}	Fig. 1	700	–	–	400	–	–	ns
Clock "High" Pulse Width	PW_{CH}	Fig. 2	700	–	–	400	–	–	ns
Clock Frequency	f_C		–	–	600	–	–	1,000	kHz
Receive Data Setup Time	t_{RDSU}	Fig. 3	350	–	–	200	–	–	ns
Receive Data Hold Time	t_{RDH}	Fig. 3	350	–	–	200	–	–	ns
Sync Match Delay Time	t_{SM}	Fig. 3	–	–	1.0	–	–	0.666	μs
Clock-to-Data Delay for Transmitter	t_{TDD}	Fig. 4,6	–	–	1.0	–	–	0.666	μs
Transmitter Underflow	t_{TUF}	Fig. 4	–	–	1.0	–	–	0.666	μs
$\overline{DTR}$ Delay Time	t_{DTR}	Fig. 5	–	–	1.0	–	–	0.666	μs
$\overline{IRQ}$ Release Time	t_{IR}	Fig. 5	–	–	1.2	–	–	0.8	μs
$\overline{RES}$ Pulse Width	t_{RES}		1.0	–	–	0.666	–	–	μs
$\overline{CTS}$ Setup Time	t_{CTS}	Fig. 6	200	–	–	150	–	–	ns
$\overline{DCD}$ Setup Time	t_{DCD}	Fig. 7	500	–	–	350	–	–	ns
Input Rise and Fall Times(Except E)	t_r, t_f	0.8V to 2.0V	–	–	1.0*	–	–	1.0*	μs

* 1.0μ or 10% of the pulse width, whichever is smaller.

2. BUS TIMING

1) READ

Item	Symbol	Test Condition	HD6852		HD68A52		Unit
			min	max	min	max	
Enable Cycle Time	t_{cycE}	Fig. 8	1.0	–	0.666	–	μs
Enable "High" Pulse Width	PW_{EH}		0.45	25	0.28	25	μs
Enable "Low" Pulse Width	PW_{EL}		0.43	–	0.28	–	μs
Setup Time, Address and R/$\overline{W}$ valid to Enable positive transition	t_{AS}		140	–	140	–	ns
Data Delay Time	t_{DDR}		–	320	–	220	ns
Data Hold Time	t_H		10	–	10	–	ns
Address Hold Time	t_{AH}		10	80	10	80	ns
Rise and Fall Time for Enable input	t_{Er}, t_{Ef}		–	25	–	25	ns

2) WRITE

Item	Symbol	Test Condition	HD6852		HD68A52		Unit
			min	max	min	max	
Enable Cycle Time	t_{cycE}	Fig. 9	1.0	–	0.666	–	μs
Enable Pulse Width, "High"	PW_{EH}		0.45	25	0.28	25	μs
Enable Pulse Width, "Low"	PW_{EL}		0.43	–	0.28	–	μs
Setup Time, Address and R/$\overline{W}$ valid to Enable positive transition	t_{AS}		140	–	140	–	ns
Data Setup Time	t_{DSW}		195	–	80	–	ns
Data Hold Time	t_H		10	–	10	–	ns
Address Hold Time	t_{AH}		10	–	10	–	ns
Rise and Fall Time for Enable input	t_{Er}, t_{Ef}		–	25	–	25	ns

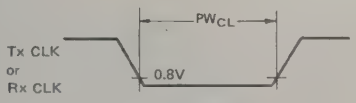


Figure 1 Clock Pulse Width ("Low" level)

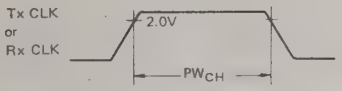


Figure 2 Clock Pulse Width ("High" level)

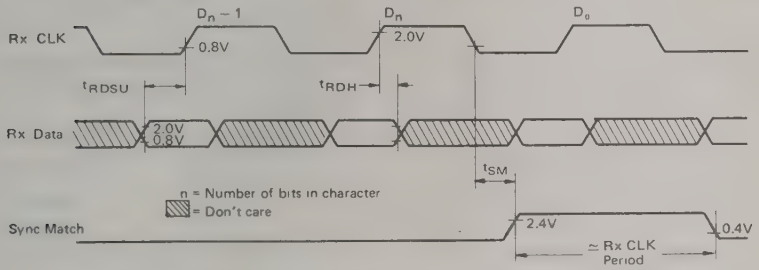


Figure 3 Receive Data Setup and Hold Times and Sync Match Delay Time

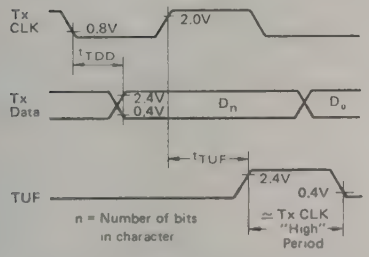
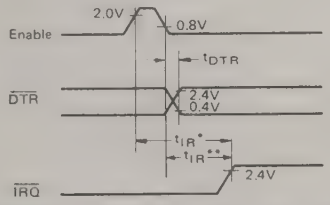


Figure 4 Transmit Data Output Delay and Transmitter Underflow Delay Time



* $\overline{IRQ}$ Release Time applied to TxData FIFO write operation and RxData FIFO read operation.

** $\overline{IRQ}$ Release Time applied to write "1" operation to RxRS, TxRS, CTUF, Clear $\overline{CTS}$ bits.

Figure 5 $\overline{DTR}$ and $\overline{IRQ}$ Release Time

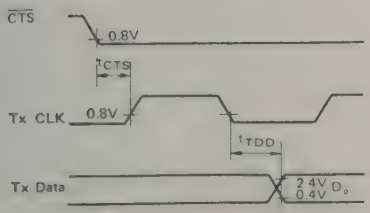
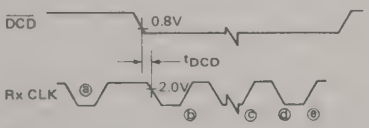


Figure 6 $\overline{CTS}$ Setup Time



Notes.

ⓐ Must occur before $\overline{DCD}$ goes low.
ⓑ First data bit placed in Rx shift register.
ⓒ Last data bit of byte placed in Rx shift register.
ⓓ Rx Data byte transferred from shift register to Rx FIFO.
ⓔ Clock edge required for generation of $\overline{IRQ}$ by RDA status.

Note: Refer to Figure 3 for the Rx Data setup and hold times.

Figure 7 $\overline{DCD}$ Setup Time

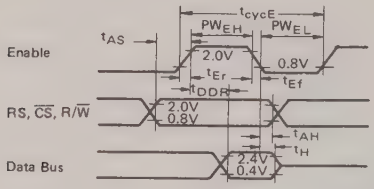


Figure 8 Bus Read Timing Characteristics (Read information from SSDA)

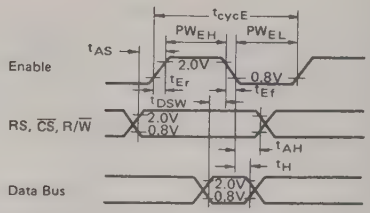


Figure 9 Bus Write Timing Characteristics (Write information into SSDA)

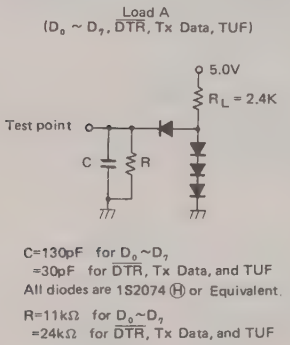


Figure 10 Test Loads

■ DEVICE OPERATION

At the bus interface, the SSDA appears as two addressable memory locations. Internally, there are seven registers: two read-only and five write-only registers. The read-only registers are Status and Receive Data; the write-only registers are Control 1, Control 2, Control 3, Sync Code and Transmit Data. The serial interface consists of serial input and output lines with independent clocks, and four peripheral/modem control lines.

Data to be transmitted is transferred directly into the 3-byte Transmit Data First-In First-Out (FIFO) Register from the data bus. Availability of the input to the FIFO is indicated by a bit in the Status Register; once data is entered, it moves through the FIFO to the last empty location. Data at the output of the FIFO is automatically transferred from the FIFO to the Transmitter Shift Register as the shift register becomes available to transmit the next character. If data is not available from the FIFO (underflow condition), the Transmitter Shift Register is automatically loaded with either a sync code or an all "1"s character. The transmit section may be programmed to append even, odd, or no parity to the transmitted word. An external control line ($\overline{CTS}$) is provided to inhibit the transmitter without clearing the FIFO.

Serial data is accumulated in the receiver based on the synchronization mode selected. In the external sync mode used for parallel-serial operation, the receiver is synchronized by the Data Carrier Detect ($\overline{DCD}$) input and transfers successive bytes of data to the input of the Receiver FIFO. The single-sync-character mode requires that a match occur between the Sync Code Register and one incoming character before data transfer to the FIFO begins. The two-sync-character mode requires that two sync codes be received in sequence to establish synchronization. Subsequent to synchronization in any mode, data is accumulated in the shift register, and parity is optionally checked. An indication of parity error is carried through the Receiver FIFO with each character to the last empty location. Availability of a word at the FIFO output is indicated by a bit in the Status Register, as is a parity error.

The SSDA and its internal registers are selected by the address bus, Read/Write (R/$\overline{W}$) and Enable control lines. To configure the SSDA, Control Registers are selected and the appropriate bits set. The Status Register is addressable for reading status.

Other I/O lines, in addition to Clear-to-Send ($\overline{CTS}$) and Data Carrier Detect ($\overline{DCD}$), include Sync Match/Data Terminal Ready (SM/$\overline{DTR}$) and Transmitter Underflow (TUF). The transmitter and receiver each have individual clock inputs allowing simultaneous operation under separate clock control. Signals to the microprocessor are the Data bus and Interrupt Request ($\overline{IRQ}$).

- **Initialization**

During a power-on sequence, the SSDA is reset via the $\overline{RES}$ input and internally latched in a reset condition to prevent erroneous output transmissions. The Sync Code Register, Control Register 2, and Control Register 3 should be programmed prior to the programmed release of the Transmitter and/or Receiver Reset bits; these bits in Control Register 1 should be cleared after the $\overline{RES}$ line has gone "High".

- **Transmitter Operation**

Data is transferred to the transmitter section in parallel form by means of the data bus and Transmit Data FIFO. The Transmit Data FIFO is a 3-byte register whose status is indicated by the Transmitter Data Register Available status bit (TDRA) and its associated interrupt enable bit. Data is transferred through the FIFO on negative edges of Enable (E) pulses. Two data transfer modes are provided in the SSDA. The 1-byte transfer mode provides for writing data to the transmitter section (and reading from the receiver section) one byte at a time. The 2-byte transfer mode provides for writing two data characters in succession.

Data will automatically transfer from the last register location in the Transmit Data FIFO (when it contains data) to the Transmitter Shift Register during the last half of the last bit of the previous character. A character is transferred into the Shift Register by the Transmitter Clock. Data is transmitted LSB first, and odd or even parity can be optionally appended. The unused bit positions in short word length characters from the data bus are "don't cares". (Note: The data bus inputs may be reversed for applications requiring the MSB to be transferred taken, e.g., IBM format for floppy disks; however, care must be taken to properly program the control registers – Table 1 will have its bit positions reversed.)

When the Shift Register becomes empty, and data is not available for transfer from the Transmit Data FIFO, an "underflow" occurs, and a character is inserted into the transmitter data stream to maintain character synchronization. The character transmitted on underflow will be either a "Mark" (all "1"s) or the contents of the Sync Code Register, depending upon the state of the Transmit Sync Code on Underflow control bit. The underflow condition is indicated by a pulse ($\approx$ Tx CLK "High" period) on the Underflow putput (when in Tx Sync on underflow mode). The Underflow output occurs coincident with the transfer of the last half of the last bit preceding the underflow character. The Underflow status bit is set until cleared by means of the Clear Underflow control bit. This output may be used in floppy disk systems to synchronize write operations and for appending CRCC.

Transmission is initiated by clearing the Transmitter Reset bit in Control Register 1. When the Transmitter Reset bit is cleared, the first full positive half-cycle of the Transmit Clock will initiate the transmit cycle, with the transmission of data or underflow characters beginning on the negative edge of the Transmit Clock pulse which started the cycle. If the Transmit Data FIFO was not loaded, an underflow character will be transmitted.

The Clear-to-Send ($\overline{CTS}$) input provides for automatic control of the transmitter by means of external system hardware; e.g., the modem $\overline{CTS}$ output provides the control in a data communications system. The $\overline{CTS}$ input resets and inhibits the transmitter section when "High", but does not reset the Transmit Data FIFO. The TDRA status bit is inhibited by $\overline{CTS}$ being "High" in either the one-sync character or two-sync-character mode of operation.

In the external sync mode, TDRA is unaffected by $\overline{CTS}$ in order to provide Transmit Data FIFO status for preloading and operating the transmitter under the control of the $\overline{CTS}$ input. When the Transmitter Reset bit (Tx Rs) is set, the Transmit Data FIFO is cleared and the TDRA status bit is cleared. After one E clock has occurred, the Transmit Data FIFO becomes available for new data with TDRA inhibited.

- **Receiver Operation**

Data and a presynchronized clock are provided to the SSDA receiver section by means of the Receive Data (Rx Data) and Receive Clock (Rx CLK) inputs. The data is a continuous stream of binary data bits without means for identifying character boundaries within the stream. It is, therefore, necessary to achieve character synchronization for the data at the beginning of the data block. Once synchronization is achieved, it is assumed to be retained for all successive characters within the block.

Data communications systems utilize the detection of sync codes during the initial portion of the preamble to establish character synchronization. This requires the detection of a single code or two successive sync codes. Floppy disk and cartridge tape units require sixteen bits of defined preamble and cassettes require eight bits of preamble to establish the reference for the start of record. All three are functionally equivalent to the detection of sync codes. Systems which do not utilize code detection techniques require custom logic external to the SSDA for character synchronization and use of the parallel-to-serial (external sync) mode.
(Note: The Receiver Shift Register is set to ones when reset)

- **Synchronization**

The SSDA provides three operating modes with respect to character synchronization: one-sync-character mode, two-sync-character mode, and external sync mode. The external sync mode requires synchronization and control of the receiving section through the Data Carrier Detect ($\overline{DCD}$) input. This external synchronization could consist of direct line control from the transmitting end of the serial data link or from external logic designed to detect the start of the message block. The one-sync-character mode searches on a bit-by-bit basis until a match is achieved between the data in the Shift Register and the Sync Code Register. The match indicates character synchronization is complete and will be retained for the message block. In the two-sync-character mode, the receiver searches for the first sync code match on a bit-by-bit basis and then looks for a second successive sync code character prior to establishing character synchronization. If the second sync code character is not received, the bit-by-bit search for the first sync code is resumed.

Sync codes received prior to the completion of synchronization (one or two character) are not transferred to the Receive Data FIFO. Redundant sync codes during the preamble or sync codes which occure as "fill characters" can automatically be stripped from the data, when the Strip Sync control bit is set, to minimize system loading. The character synchronization will be retained until cleared by means of the Clar Sync bit, which also inhibits synchronization search when set.

- **Receiving Data**

Once synchronization has been achieved, subsequent characters are automatically transferred into the Receive Data FIFO and clocked through the FIFO to the last empty location by E pulses (MPU System $\phi 2$). The Receiver Data Available status bit

(RDA) indicates when data is available to be read from the last FIFO location (#3) when in the 1-byte transfer mode. The 2-byte transfer mode causes the RDA status bit to indicate data is available when the last two FIFO register locations are full. Data being available in the Receive Data FIFO causes an interrupt request if the Receiver Interrupt Enable (RIE) bit is set. The MPU will then read the SSDA Status Register, which will indicate that data is available for the MPU read from the Receiver Data FIFO register. The $\overline{IRQ}$ and RDA status bits are reset by a read from the FIFO. If more than one character has been received and is resident in the Receive Data FIFO, subsequent E clocks will cause the FIFO to update and the RDA and $\overline{IRQ}$ status bits will again be set. The read data operation for the 2-byte transfer mode requires an intervening E clock between reads to allow the FIFO data to shift. Optional parity is automatically checked as data is received, and the parity status condition is maintained with each character until the data is read from the Receive Data FIFO. Parity errors will cause an interrupt request if the Error Interrupt Enable (EIE) has been set. The parity bit is not transferred to the data bus but must be checked in the Status Register. NOTE: In the 2-byte transfer mode, parity should be checked prior to reading the second byte, since a FIFO read clears the error bit.

Other status bits which pertain to the receiver section are Receiver Overrun and Data Carrier Detect ($\overline{DCD}$). The Overrun status bit is automatically set when a transfer of a character to the Receive Data FIFO occurs and the first register of the Receive Data FIFO is full. Overrun causes an interrupt if Error Interrupt Enable (EIE) has been set. The transfer of the overrunning character into the FIFO causes the previous character in the FIFO input register location to be lost. The Overrun status bit is cleared by reading the Status Register (when the overrun condition is present), followed by a Receive Data FIFO Register read. Overrun cannot occur and be cleared without providing an opportunity to detect its occurrence via the Status Register.

A positive transition on the $\overline{DCD}$ input causes an interrupt if the EIE control bit has been set. The interrupt caused by $\overline{DCD}$ is cleared by reading the Status Register when the $\overline{DCD}$ status bit is "1", followed by a Receive Data FIFO read. The $\overline{DCD}$ status bit will subsequently follow the state of the $\overline{DCD}$ input when it goes "Low".

■ SSDA REGISTERS

Seven registers in the SSDA can be accessed by means of the bus. The registers are defined as read-only or write-only according to the direction of information flow. The Register Select (RS) input selects two registers in each state, one being read-only and the other write-only. The Read/Write (R/$\overline{W}$) input defined which of the two selected registers will actually be accessed. Four registers (two read-only and two write-only) can be addressed via the bus at any particular time. These registers and the required addressing are defined in Table 1.

● Control Register 1 (C1)

Control Register 1 is an 8-bit wirte-only register that can be directly addressed from the data bus. Control Register 1 is addressed when RS = "Low" and R/$\overline{W}$ = "Low".

Receiver Reset (Rx Rs), C1 Bit 0

The Receiver Reset control bit provides both a reset and inhibit function to the receiver section. When Rx Rs is set, it clears the receiver control logic, error logic, Rx Data FIFO Control, Parity Error status bit, and $\overline{DCD}$ interrupt. The Receiver Shift Register is set ones. The Rx Rs bit must be cleared after the occurrence of a "Low" level on $\overline{RES}$ in order to enable the receiver section of the SSDA.

Transmitter Reset (Tx Rs), C1 Bit 1

The Transmitter Reset control bit provides both a reset and inhibit to the transmitter section. When Tx Rs is set, it clears the transmitter control section, Transmitter Shift Register, Tx Data FIFO Control (the Tx Data FIFO can be reloaded after one E clock pulse), the Transmitter Underflow status bit, and the $\overline{CTS}$ interrupt, and inhibits the TDRA status bit (in the one-sync-character and two-sync-character modes). The Tx Rs bit must be cleared after the occurrence of a "Low" level on $\overline{RES}$ in order to enable the transmitter section of the SSDA. If the Tx FIFO is not preloaded, it must be loaded immediately after the Tx Rs release to prevent a transmitter underflow condition.

Strip Synchronization Characters (Strip Sync), C1 Bit 2

If the Strip Sync bit is set, the SSDA will automatically strip all received characters which match the contents of the Sync Code Register. The characters used for synchronization (one or two characters of sync) are always stripped from the received data stream.

Clear Synchronization (Clear Sync), C1 Bit 3

The Clear Sync control bit provides the capability of dropping receiver character synchronization and inhibiting resynchronization. The Clear Sync bit is set to clear and inhibit receiver synchronization in all modes and is reset to zero to enable resynchronization.

Transmitter Interrupt Enable (TIE), C1 Bit 4

TIE enable both the Interrupt Request ($\overline{IRQ}$) output and Interrupt Request status bit to indicate a transmitter service request. When TIE is set and the TDRA status bit is "1", the $\overline{IRQ}$ output will go "Low" (the active state) and the $\overline{IRQ}$ status bit will go "1".

Receiver Interrupt Enable (RIE), C1 Bit 5

RIE enable both the Interrupt Request output ($\overline{IRQ}$) and the Interrupt Request status bit to indicate a receiver service request. When RIE is set and the RDA status bit is "1", the $\overline{IRQ}$ output will go "Low" (the active state) and the $\overline{IRQ}$ status bit will go "1".

Address Control 1 (AC1) and Address Control 2 (AC2), C1 Bits 6 and 7

AC1 and AC2 select one of the write-only registers – Control 2, Control 3, Sync Code, or Tx Data FIFO – as shown in Table 1, when RS = "High" and R/$\overline{W}$ = "Low".

● Control Register 2 (C2)

Control Register 2 is an 8-bit write-only register which can be programmed from the bus when the Address Control bits in Control Register 1 (AC1 and AC2) are reset, RS = "High" and R/$\overline{W}$ = "Low".

Peripheral Control 1 (PC1) and Peripheral Control 2 (PC2), C2 Bits 0 and 1

Two control bits, PC1 and PC2, determine the operating characteristics of the Sync Match/$\overline{DTR}$ output. PC1, when "High", selects the Sync Match mode. PC2 provides the inhibit/

enable control for the SM/$\overline{DRT}$ output in the Sync Match mode. A one-bit-wide pulse is generated at the output when PC2 is "0", and a match occurs between the contents of the Sync Code Register and the incoming data even if sync is inhibited (Clear Sync bit = "1"). The Sync Match pulse is referenced to the negative edge of Rx CLK pulse causing the match.

The Data Terminal Ready ($\overline{DTR}$) mode is selected when PC1 is "0". When PC2 = "1" the SM/$\overline{DTR}$ output = "Low" and vice versa. The operation of PC2 and PC1 is summarized in Table 4.

1-Byte/2-Byte Transfer (1-Byte/2-Byte), C2 Bit 2

When 1-Byte/2-Byte is set, the TDRA and RDA status bits will indicate the availability if their respective data FIFO registers for a single byte data transfer. Alternately, if 1-Byte/2-Byte is reset, the TDRA and RDA status bits indicate when two bytes of data can be moved without a second status read. An intervening Enable pulse must occur between data transfers.

Word Length Selects (WS1, WS2, WS3), C2 Bits 3, 4, 5

Word length Select bits WS1, WS2, and WS3 select word length of 7, 8, or 9 bits including parity as shown in Table 3.

Transmit Sync Code on Underflow (Tx Sync), C2 Bit 6

When Tx Sync is set, the transmitter will automatically send a sync character when data is not available for transmission. If Tx Sync is reset, the transmitter will transmit a Mark character (including the parity bit position) on underflow. When the underflow is detected, a pulse approximately a Tx CLK "High" period wide will occur on the underflow output if the Tx Sync bit is "1". Internal parity generation is inhibited during underflow except for sync code fill character transmission in 8 bit plus parity word lengths.

Error Interrupt Enable (EIE), C2 Bit 7

When EIE is set, the $\overline{IRQ}$ status bit will go "1" and the $\overline{IRQ}$ output will go "Low" if:

1) A receiver overrun occurs. The interrupt is cleared by reading the Status Register and reading the Rx Data FIFO.
2) $\overline{DCD}$ input has gone to a "High". The interrupt is cleared by reading the Status Register and reading the Rx Data FIFO.
3) A parity error exists for the character in the last location (#3) of the Rx Data FIFO. The interrupt is cleared by reading the Rx Data FIFO. The interrupt is cleared by reading the Rx Data FIFO.
4) The $\overline{CTS}$ input has gone to a "High". The interrupt is cleared by writing a "1" in the Clear $\overline{CTS}$ bit, C3 bit 2, or by a Tx Reset.
5) The transmitter has underflowed (in the Tx Sync on Underflow mode). The interrupt is cleared by writing a "1" into the Clear Underflow, C3 bit 3, or Tx Reset.

When EIE is a "0", the $\overline{IRQ}$ status bit and the $\overline{IRQ}$ output are disabled for the above error conditions. A "Low" level on the $\overline{RES}$ input resets EIE to "0".

- **Control Register 3 (C3)**

Control Register 3 is a 4-bit write-only register which can be programmed from the bus when RS = "High" and R/$\overline{W}$ = "Low" and Address Control bit AC1 = "1" and AC2 = "0".

External/Internal Sync Mode Control (E/I Sync), C3 Bit 0

When the E/I Sync Mode bit is "1", the SSDA is in the external sync mode and the receiver synchronization logic is disabled. Synchronization can be achieved by means of the $\overline{DCD}$ input or by starting Rx CLK at the midpoint of data bit "0" of a character with $\overline{DCD}$ "Low". Both the transmitter and receiver sections operate as parallel – serial converters in the External Sync mode. The Clear Sync bit in Control Register 1 acts as a receiver sync inhibit when "High" to provide a bus controllable inhibit. The Sync Code Register can serve as a transmitter fill character register and a receiver match register in this mode. A "Low" on the $\overline{RES}$ input resets the E/I Sync Mode bit placing the SSDA In the internal sync mode.

One-Sync-Character/Two-Sync-Character Mode, Control (1 Sync/2 Sync), C3 Bit 1

When the 1 Sync/2 Sync bit is set, the SSDA will synchronize on a single match between the received data and the contents of the Sync Code Register. When the 1 Sync/2 Sync bit is reset, two successive sync characters must be received prior to receiver synchronization. If the second sync character is not detected, the bit by bit search resumes from the first bit in the second character. See the description of the Sync Code Register for more details.

Clear $\overline{CTS}$ Status (Clear $\overline{CTS}$), C3 Bit 2

When a "1" is written into the Clear $\overline{CTS}$ bit, the stored status and interrupt are cleared. Subsequently, the $\overline{CTS}$ status bit reflects the state of the $\overline{CTS}$ input. The Clear $\overline{CTS}$ control bit does not affect the $\overline{CTS}$ input nor its inhibit of the transmitter secton. The Clear $\overline{CTS}$ command bit is self-clearing, and writing a "0" into this bit is a nonfunctional operation.

Clear Transmit Underflow Status (CTUF), C3 Bit 3

When a "1" is written into the CTUF status bit, the CTUF bit and its associated interrupt are reset. The CTUF command bit is self-clearing and writing a "0" into this bit is a nonfunctional operation.

- **Sync Code Register**

The Sync Code Register is an 8-bit register for storing the programmable sync code required for received data character synchronization in the one-sync-character and two-sync-character modes. The Sync Code Register also provides for stripping the sync/fill characters from the received data (a programmable option) as well as automatic insertion of fill characters in the transmitted data stream. The Sync Code Register is not utilized for teceiver character synchronization in the external sync mode; however, it provides storage of receiver match and transmit fill characters.

The Sync Code Register can be loaded when AC2 and AC1 are a "1" and "0" respectively, and R/$\overline{W}$ = "Low" and RS = "High".

The Sync Code Register may be changed after the detection of a match with the received data (the first sync code having been detected) to synchronize with a double-word sync pattern. (This sync code change must occur prior to the completion of the second character.) The sync match (SM) output can be used to interrupt the MPU system to indicate that the first eight bits have matched. The service routine would then change the sync match register to the second half of the pattern. Alternately, the one-sync-character mode can be used for sync codes for 16 or more bits by using software to check the second and subsequent bytes after reading them from the FIFO.

The detection of the sync code can be programmed to appear on the Sync Match/$\overline{DTR}$ output by writing a "1" in PC1 (C2 bit 0) and a "0" in PC2 (C2 bit 1). The Sync Match output will go "High" for one bit time beginning at the character interface between the sync code and the next character.

- **Parity for Sync Character**

Transmitter

Transmitter does not generate parity for the sync character except 9-bit mode.

9-bit (8-bit + parity) – 8-bit sync character + parity
8-bit (7-bit + parity) – 8-bit sync character (no parity)
7-bit (6-bit + parity) – 7-bit sync character (no parity)

Receiver

At Synchronization

Receiver automatically strips the sync character(s) (two sync characters if '2 sync' mode is selected) which is used to establish synchronization. And parity is not checked for these sync characters.

After Synchronization is Established

When 'strip sync' bit is selected, the sync characters (fill characters) are stripped and parity is not checked for the stripped sync (fill) characters. When 'strip sync' bit is not selected (0), the sync character is assumed to be normal data and it is transferred into FIFO after parity checking. (When non-parity format is selected, parity is not checked.)

Strip Sync (C1 Bit 2)	Data Format (C2 Bit 3-5)	Operation
1	x	No transfer of sync code. No parity check of sync code.
0	With Parity	*Transfer data and sync codes. Parity check.
0	Without Parity	*Transfer data and sync codes. No parity check.

* Subsequent to synchronization
x ..... don't care

It is necessary to pay attention to the selected sync character in the following cases.

1) Data format is (6 + parity), (7 + parity),
2) Strip sync is not selected ("0").
3) After synchronization when sync code is used as a fill character.

Transmitter sends sync character without parity, but receiver checks the parity as if it is normal data. Therefore, the sync character should be chosen to match the parity check selected for the receiver in this special case.

- **Receive Data First-In First-Out Register (Rx Data FIFO)**

The Receive Data FIFO Register consists of three 8-bit registers which are used for buffer storage of received data. Each 8-bit register has an internal status bit which monitors its full or empty condition. Data is always transferred from a full register to an adjacent empty register. The transfer from register to register occurs on E pulses. The RDA status bit will be "1" when data is available in the last location of the Rx Data FIFO.

In an Overrun condition, the overrunning character will be transferred into the full first stage of the FIFO register and will cause the loss of that data character. Successive overruns continue to overwrite the first register of the FIFO. This destruction of data is indicated by means of the Overrun status bit. The Overrun bit will be set when the overrun occurs and remains set until the Status Register is read, followed by a read of the Rx Data FIFO.

Unused data bits for short word lengths (including the parity bit) will appear as "0"s on the data bus when the Rx Data FIFO is read.

- **Transmit Data First-In First-Out Register (Tx Data FIFO)**

The Transmit Data FIFO Register consists of three 8-bit registers which are used for buffer storage of data to be transmitted. Each 8-bit register has an internal status bit which monitors its full or empty condition. Data is always transferred from a full register to an adjacent empty register. The transfer is clocked by E pulses.

The TDRA status bit will be "High" if the Tx Data FIFO is available for data.

Unused data bits for short word lengths will be handled as "don't cares". The parity bit is not transferred over the data bus since the SSDA generates parity at transmission.

When an Underflow occurs, the Underflow character will be either the contents of the Sync Code Register or an all "1"s character. The underflow will be stored in the Status Register until cleared and will appear on the Underflow output as a pulse approximately a Tx CLK "High" period wide.

- **Status Register**

The Status Register is an 8-bit read-only register which provides the real-time status of the SSDA and the associated serial data channel. Reading the Status Register is a non-destructive process. The method of clearing status bits depends upon the function each bit represents and is discussed for each bit in the register.

Receiver Data Available (RDA), S Bit 0

The Receiver Data Available status bit indicates when receiver data can be read from the Rx Data FIFO. The receiver data being present in the last register (#3) of the FIFO causes RDA to be "1" for the 1-byte transfer mode. The RDA bit being "1" indicates that the last two registers (#2 and #3) are full when in the 2-byte transfer mode. The second character can be read without a second status rad (to determine that the character is available). And E pulse must occur between reads of the Rx Data FIFO to allow the FIFO to shift. Status must be read on a word-by-word basis if receiver data error checking is important. The RDA status bit is reset automatically when data is not available.

Transmitter Data Register Available (TDRA), S Bit 1

The TDRA status bit indicates that data can be loaded into the Tx Data FIFO Register. The first register (#1) of the Tx Data FIFO being empty will be indicated by a "1" in the TDRA status bit in the 1-byte transfer mode. The first two registers (#1 and #2) must be empty for TDRA to be "1" when in the 2-byte transfer mode. The Tx Data FIFO can be loaded with two bytes without an intervening status read; however, one E pulse must occur between loads. TDRA is inhibited by the Tx Reset or $\overline{RES}$. When Tx Reset is set, the Tx Data FIFO is cleared and then released on the next E clock pulse. The Tx Data FIFO can then be loaded with up to three characters of data, even though TDRA is inhibited. This feature allows preloading data prior to the release of Tx Reset. A "High" level on the $\overline{CTS}$ input inhibits the TDRA status bit in either sync mode of operation (one-sync-character or two-sync-character). $\overline{CTS}$ does not affect TDRA in the external sync mode. This

enables the SSDA to operate under the control of the $\overline{CTS}$ input with TDRA indicating the status of the Tx Data FIFO. The $\overline{CTS}$ input does not clear the Tx Data FIFO in any operating mode.

Data Carrier Detect ($\overline{DCD}$), S Bit 2

A positive transition on the $\overline{DCD}$ input is stored in the SSDA until cleared by reading both Status and Rx Data FIFO. A "1" written into Rx Rs also clears the stored $\overline{DCD}$ status. The $\overline{DCD}$ status bit, when set, indicates that the $\overline{DCD}$ input has gone "High", The reading of both Status and Receive Data FIFO allows Bit 2 of subsequent Status reads to indicate the state of the $\overline{DCD}$ input until the next positive transition.

Clear-to-Send ($\overline{CTS}$), S Bit 3

A positive transiton on the $\overline{CTS}$ input is stored in the SSDA until cleared by writing a "1" into the Clear $\overline{CTS}$ control bit or the Tx Rs bit. The $\overline{CTS}$ status bit, when set, indicates that the $\overline{CTS}$ input has gone "High". The Clear $\overline{CTS}$ command (a "1" into C3 Bit 2) allows Bit 3 of subsequent Status reads to indicate the state of the $\overline{CTS}$ input until the next positive transition.

Transmitter Underflow (TUF), S Bit 4

When data is not available for the transmitter, an underflow occurs and is so indicated in the Status Register (in the Tx Sync on underflow mode). The underflow status bit is cleared by writing a "1" into the Clear Underflow (CTUF) control bit or the Tx Rs bit. TUF indicates that a sync character will be transmitted as the next character. A TUF is indicated on the output only when the contents of the Sync Code Register is to be transferred (transmit sync code on underflow = "1").

Receiver Overrun (Rx Ovrn), S Bit 5

Overrun indicates data has been received when the Rx Data FIFO is full, resulting in data loss. The Rx Ovrn status bit is set when Overrun occurs. The Rx Ovrn status bit is cleared by reading Status followed by reading the Rx Data FIFO or by setting the Rx Rs control bit.

Receiver Parity Error (PE), S Bit 6

The parity error status bit indicates that parity for the character in the last register of the Rx Data FIFO did not agree with selected parity. The parity error is cleared when the character to which it pertains is read from the Rx Data FIFO or when Rx Rs occurs. The $\overline{DCD}$ input does not clear the Parity Error or Rx Data FIFO status bits.

Interrupt Request ($\overline{IRQ}$), S Bit 7

The Interrupt Request status bit indicates when the $\overline{IRQ}$ output is in the active state ($\overline{IRQ}$ output = "Low"). The IRQ status bit is subject to the same interrupt enables (RIE, TIE, and EIE) as the $\overline{IRQ}$ output. The IRQ status but simplifies status inquiries for polling systems by providing single bit indication of service requests.

Table 1 SSDA Programming Model

Register	Control* Inputs		Address Control		Register Content							
	RS	R/$\overline{W}$	AC2	AC1	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Status (S)	0	1	X	X	Interrupt Request ($\overline{IRQ}$)	Receiver Parity Error (PE)	Receiver Overrun (Rx Ovrn)	Transmitter Underflow (TUF)	Clear-to-Send ($\overline{CTS}$)	Data Carrier Detect ($\overline{DCD}$)	Transmitter Data Register Available (TDRA)	Receiver Data Available (RDA)
Control 1 (C1)	0	0	X	X	Address Control 2 (AC2)	Address Control 1 (AC1)	Receiver Interrupt Enable (RIE)	Transmitter Interrupt Enable (TIE)	Clear Sync	Strip Sync Characters (Strip Sync)	Transmitter Reset (Tx Rs)	Receiver Reset (Rx Rs)
*** Receive Data FIFO	1	1	X	X	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
Control 2 (C2)	1	0	0	0	Error Interrupt Enable (EIE)	Transmit Sync Code on Underflow (Tx Sync)	Word Length Select 3 (WS3)	Word Length Select 2 (WS2)	Word Length Select 1 (WS1)	1-Byte/2-Byte Transfer (1-Byte/2-Byte)	Peripheral Control 2 (PC2)	Peripheral Control 1 (PC1)
Control 3 (C3)	1	0	0	1	Not Used	Not Used	Not Used	Not Used	Clear Transmitter Underflow Status (CTUF)	Clear $\overline{CTS}$ Status (Clear $\overline{CTS}$)	One-Sync-Character/ Two-Sync Character Mode Control (1 Sync/ 2 Sync)	External/ Internal Sync Mode Control (E/I Sync)
** *** Sync Code	1	0	1	0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
*** Transmit Data FIFO	1	0	1	1	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0

* 0 ; "Low" level, 1 ; "High" level

** "FF" should not be used as Sync Code.

*** When the SSDA is used in applications requiring the MSB of data to be receive and transmitted first, the data bus inputs to the SSDA may be reversed (D_0 to D_7, etc.) Caution must be used when this is done since the bit positions in this table will be reversed, and the parity should not be selected.

Table 2 Functions of SSDA Register

Register	Bit	Symbol	Function			
	7	$\overline{IRQ}$	The $\overline{IRQ}$ flag is cleared when the source of the $\overline{IRQ}$ is cleared. The source is determined by the enables in the Control Registers: TIE, RIE, EIE.			
Status Register (S)	6	PE	Conditions for Set	When parity error is detected in receive data.	Conditions for Reset	Read Rx Data FIFO, or a "1" into Rx Rs (C1 Bit 0).
	5	Rx Ovrn		When receive data FIFO overruns.		Read Status and then Rx Data FIFO, or a "1" into Rx Rs (C1 Bit 0).
	4	TUF		When under flow is occurred in the transmitter.		A "1" into CTUF (C3 Bit 3) or into Tx Rs (C1 Bit 1).
	3	$\overline{CTS}$		When $\overline{CTS}$ signal rises.		A "1" into Clear $\overline{CTS}$ (C3 Bit 2) or a "1" into Tx Rs (C1 Bit 1)
	2	$\overline{DCD}$		When $\overline{DCD}$ signal rises.		Read Status and then Rx Data FIFO or a "1" into Rx Rs (C1 Bit 0)
	1	TDRA		1 Byte Transfer Mode; when the transmit data FIFO (#1) is empty.		Write into Tx Data FIFO.
				2 Byte Transfer Mode; when the transmit data FIFO (#1, #2) is empty.		
	0	RDA		1 Byte Transfer Mode; when the data is received in the receive data FIFO (#3).		Read Rx Data FIFO.
				2 Byte Transfer Mode; when the data is received in the receive data FIFO (#2, #3).		
Control Register 1 (C1)	7 6	AC2 AC1	Used to access other registers, as shown Table 1.			
	5	RIE	When "1", enables interrupt on RDA (S Bit 0).			
	4	TIE	When "1", enables interrupt on TDRA (S Bit 1).			
	3	Clear Sync	When "1", clears receiver character synchronization.			
	2	Strip Sync	When "1", strips all sync codes from the received data stream.			
	1	Tx Rs	When "1", resets and inhibits the transmitter section.			
	0	Rx Rs	When "1", resets and inhibits the receiver section.			
Control Register 2 (C2)	7	EIE	When "1", enables the PE, Rx Ovrn, TUF, $\overline{CTS}$, and $\overline{DCD}$ interrupt flags (S Bits 6 through 2).			
	6	Tx Sync	When "1", allows sync code contents to be transferred on underflow, and enables the TUF Status bit and output. When "0", an all mark character is transmitted on underflow.			
	5 4 3	WS3 WS2 WS1	Word Length Select			
	2	1-Byte/2-Byte	When "1", enables the TDRA and RDA bits to indicate when a 1-byte transfer can occur; when "0", the TDRA and RDA bits indicate when a 2-byte transfer can occur.			
	1 0	PC2 PC1	SM/$\overline{DTR}$ Output Control			
Control Register 3 (C3)	3	CTUF	When "1", clears TUF (S Bit 4), and $\overline{IRQ}$ if enabled.			
	2	Clear $\overline{CTS}$	When "1", clears $\overline{CTS}$ (S Bit 3), and $\overline{IRQ}$ if enabled.			
	1	1-Sync/2-Sync	When "1", selects the one-sync-character mode; when "0", selects the two-sync-character mode.			
	0	E/I Sync	When "1", selects the external sync mode; when "0", selects the internal sync mode.			

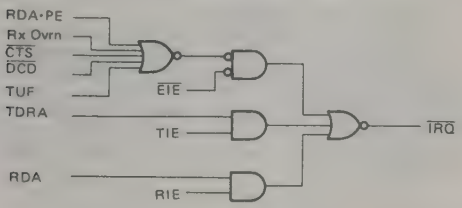
Table 3 Word Length

Bit 5 WS3	Bit 4 WS2	Bit 3 WS1	Word Length
0	0	0	6 Bits + Even Parity
0	0	1	6 Bits + Odd Parity
0	1	0	7 Bits
0	1	1	8 Bits
1	0	0	7 Bits + Even Parity
1	0	1	7 Bits + Odd Parity
1	1	0	8 Bits + Even Parity
1	1	1	8 Bits + Odd Parity

Table 4 SM/$\overline{DTR}$ Output Control

Bit 1 PC2	Bit 0 PC1	SM/$\overline{DTR}$ Output at Pin 5
0	0	"High" Level*
0	1	Pulse ⎍ 1-Bit Wide, on SM
1	0	"Low" Level*
1	1	SM Inhibited, "Low"*

* OUTPUT level is fixed by the data written into PC2, PC1.
** When "10" or "11", output is fixed at "Low".



■ INTERFACE SIGNALS FOR MPU

The SSDA interfaces to the HD6800 MPU with an 8-bit bi-directional data bus, a chip select line, a register select line, an interrupt request line, read/write line, an enable line, and a reset line. These signals, in conjunction with the HD6800 VMA output, permit the MPU to have complete control over the SSDA.

● Bi-Directional Data Bus (D_0~D_7)

The bi-directional data bus (D_0~D_7) allow for data transfer between the SSDA and the MPU. The data bus output drivers are three-state devices that remain in the high impedance (off) state except when the MPU performs an SSDA read operation.

● Enable (E)

The Enable signal, E, is a high impedance TTL compatible input that enables the bus input/output data buffers, clocks data to and from the SSDA, and moves data through the FIFO Registers. This signal is normally the continuous HMCS6800 System $\phi 2$ clock, so that incoming data characters are shifted through the FIFO.

● Read/Write (R/$\overline{W}$)

The Read/Write line is a high impedance input that is TTL compatible and is used to control the direction of data flow through the SSDA's input/output data bus interface. When Read/Write is "High" (MPU read cycle), SSDA output drivers are turned on if the chip is selected and a selected register is read. When it is "Low", the SSDA output drivers are turned off and the MPU writes into a selected register. The Read/Write signal is also used to select read-only or write-only registeres within the SSDA.

● Chip Select ($\overline{CS}$)

This high impedance TTL compatible input line is used to address the SSDA. The SSDA is selected when $\overline{CS}$ is "Low". VMA should be used in generating the $\overline{CS}$ input to insure that false selects will not occur. Transfers of data to and from the SSDA are then performed under the control of the Enable signal, Read/Write, and Register Select.

● Register Select (RS)

The Register Select line is a high impedance input that is TTL compatible. A "High" level is used to select Control Registers C2 and C3, the Sync Code Register, and the Transmit/Receive Data Registers. A "Low" level selects the Control 1 and Status Registers (see Table 1).

● Interrupt Request ($\overline{IRQ}$)

$\overline{IRQ}$ is a TTL compatible, open-drain (no internal pullup), active "Low" output that is used to interrupt the MPU. The $\overline{IRQ}$ remains "Low" until cleared by the MPU.

● Reset ($\overline{RES}$)

The $\overline{RES}$ input provides a means of resetting the SSDA from an external source. In the "Low" state, the $\overline{RES}$ input causes the following:

1) Receiver Reset (Rx Rs) and Transmitter Reset (Tx Rs) bits are set causing both the receiver and transmitter sections to be held in a reset condition.
2) Peripheral Control bits PC1 and PC2 are reset to zero, causing the SM/$\overline{DTR}$ output to be "High".
3) The Error Interrupt Enable (EIE) bit is reset.
4) An internal synchronization mode is selected.
5) The Transmitter Data Register Available (TDRA) status bit is cleared and inhibited.

When $\overline{RES}$ returns "High" (the inactive state), the transmitter and receiver sections will remain in the reset state until the Receiver Reset and Transmitter Reset bits are cleared via the bus under software control. The control Register bits affected by $\overline{RES}$ (Rx Rs, Tx Rs, PC1, PC2, EIE, and E/I Sync) cannot be changed when $\overline{RES}$ is "Low".

■ CLOCK INPUTS

Separate high impedance TTL compatible inputs are provided for clocking of transmitted and received data.

● Transmit Clock (Tx CLK)

The Transmit Clock input is used for the clocking of transmitted data. The transmitter shifts data on the negative transition of the clock.

● Receive Clock (Rx CLK)

The Receive Clock input is used for clocking in received data. The clock and data must be synchronized externally. The receiver samples the data on the positive transition of the clock.

■ SERIAL INPUT/OUTPUT LINES

● Receive Data (Rx Data)

The Receive Data line is a high impedance TTL compatible input through which data is received in a serial format. Data rates are from 0 to 600 kbps.

● Transmit Data (Tx Data)

The Transmit Data output line transfers serial data to a modem or other peripheral. Data rates are from 0 to 600 kbps.

■ PERIPHERAL/MODEM CONTROL

The SSDA includes several functions that permit limited control of a peripheral or modem. The functions included are $\overline{CTS}$, SM/$\overline{DTR}$, $\overline{DCD}$, and TUF.

● Clear-to-Send ($\overline{CTS}$)

The $\overline{CTS}$ input provides a real-time inhibit to the transmitter

section (the Tx Data FIFO is not disturbed). A positive $\overline{CTS}$ transition resets the Tx Shift Register and inhibits the TDRA status bit and its associated interrupt in both the one-sync-character and two-sync-character modes of operation. TDRA is not affected by the $\overline{CTS}$ input in the external sync mode.

The positive transition of $\overline{CTS}$ is stored within the SSDA to insure that its occurrence will be acknowledged by the system. The stored $\overline{CTS}$ information and its associated $\overline{IRQ}$ (if enabled) are cleared by writing a "1" in the Clear $\overline{CTS}$ bit. The $\overline{CTS}$ status bit subsequently follows the $\overline{CTS}$ input when it goes "Low".

The $\overline{CTS}$ input provides character timing for transmitter data when in the external sync mode. Transmission is initiated on the negative transition of the first full positive clock pulse of the transmitter clock (Tx CLK) after the release of $\overline{CTS}$ (see Figure 6).

- **Data Carrier Detect ($\overline{DCD}$)**

The $\overline{DCD}$ input provides a real-time inhibit to the receiver section (the Rx FIFO is not disturbed). A positive $\overline{DCD}$ transition resets and inhibits the receiver section except for the Receive FIFO and the RDRA status bit and its associated $\overline{IRQ}$.

The positive transition of $\overline{DCD}$ is stored within the SSDA to insure that its occurrence will be acknowledged by the system. The stored $\overline{DCD}$ information and its associated $\overline{IRQ}$ (if enabled) are cleared by reading the Status Register and then the Receiver FIFO, or by writing a "1" into the Receiver Reset bit. The $\overline{DCD}$ status bit subsequently follows the $\overline{DCD}$ input when it goes "Low". The $\overline{DCD}$ input provides character synchronization timing for the receiver during the external sync mode of operation. The receiver will be initialized and data will be sampled on the positive transition of the first full Receive Clock cycle after release of $\overline{DCD}$ (see Figure 7).

- **Sync Mach/Data Terminal Ready (SM/$\overline{DTR}$)**

The SM/$\overline{DTR}$ output provides four functions (see Table 4) depending on the state of the PC1 and PC2 control bits. When the Sync Match mode is selected (PC1 = "1", PC2 = "0"), the output provides a one-bit-wide pulse when a sync code is detected. This pulse occurs for each sync code match even if the receiver has already attained synchronization. The SM output is inhibited when PC2 = "1". The $\overline{DTR}$ mode (PC1 = "0") provides an output level corresponding to the complement of PC2 ($\overline{DTR}$ = "0" when PC2 = "1".) (see Table 4.)

- **Transmitter Underflow (TUF)**

The Underflow output indicates the occurrence of a transfer of a "fill character" to the Transmitter Shift Register when the last location (#3) in the Transmit Data FIFO is empty. The Underflow output pulse is approximately a Tx CLK "High" period wide and occurs during the last half of the last bit of the character preceding the "Underflow" (see Figure 4). The Underflow output pulse does not occur when the Tx Sync bit is in the reset state.

- **NOTE FOR USAGE**

If the hold time of $\overline{CS}$ signal and R/$\overline{W}$ signal is within 50~230 ns, there is a case that Transmit Data FIFO is not cleared and TDRA flag is not set when software reset using TxRS (TxRS=1) is executed. Usual program for data transmission will start to send the data as shown in Fig. 11 and Fig. 12.

In this case, the data of the first three bytes are not preset and unexpected data which is remaining in Transmit Data FIFO are sent in the first two bytes.

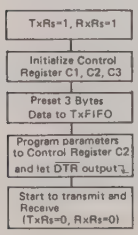

TxRs=1, RxRs=1
Initialize Control Register C1, C2, C3
Preset 3 Bytes Data to TxFIFO
Program parameters to Control Register C2 and let DTR output "L"
Start to transmit and Receive (TxRs=0, RxRs=0)

Figure 11 Normal Flow of Starting the Transmission and Reception

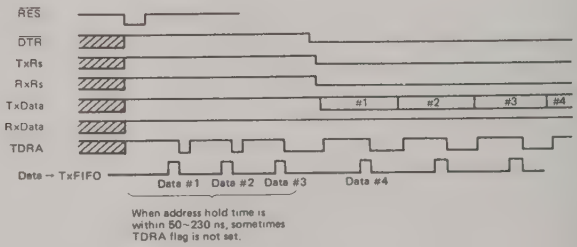


Figure 12 Transmission Start Sequence

In case of SSDA, Address Hold Time should be from 20 to 50 ns or over 230 ns.

- **$\overline{DCD}$ Input in External Synchronization Mode**

In case of receiving data in External Synchronization Mode, Receive data is put off by one bit at times, when $\overline{DCD}$ is drived like ⌐└ in RxCLK cycle in which RDA flag is set.

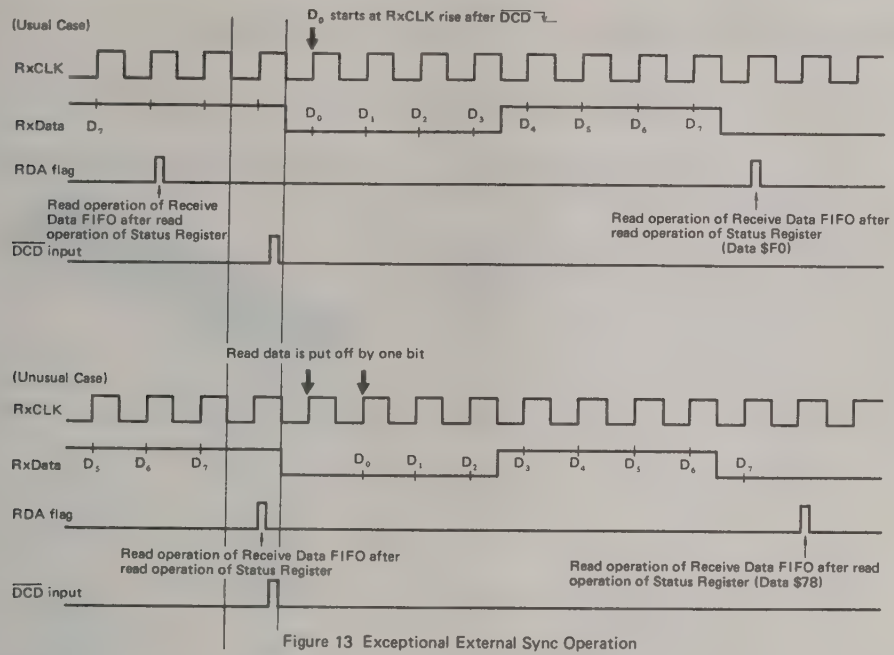

(Usual Case)
D0 starts at RxCLK rise after DCD
RxCLK
RxData
D7
D0 D1 D2 D3 D4 D5 D6 D7
RDA flag
Read operation of Receive Data FIFO after read operation of Status Register
DCD input
Read operation of Receive Data FIFO after read operation of Status Register (Data $F0)
Read data is put off by one bit
(Unusual Case)
RxCLK
RxData
D5 D6 D7
D0 D1 D2 D3 D4 D5 D6 D7
RDA flag
Read operation of Receive Data FIFO after read operation of Status Register
DCD input
Read operation of Receive Data FIFO after read operation of Status Register (Data $78)

Figure 13 Exceptional External Sync Operation

To avoid this case, use SSDA in the following method.

(1) $\overline{DCD}$ ↓ and RxCLK ↓ should meet the relation shown in Fig. 14.

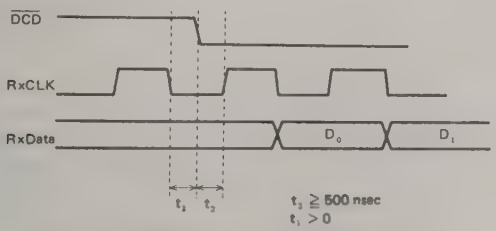


Figure 14 $\overline{DCD}$ Input Timing in External Sync Mode

(2) RxData should be input regarding the second RxCLK rise as D_0 bit, after $\overline{DCD}$ ↓.

HD46508, HD46508-1, HD46508A, HD46508A-1

ADU (Analog Data Acquisition Unit)

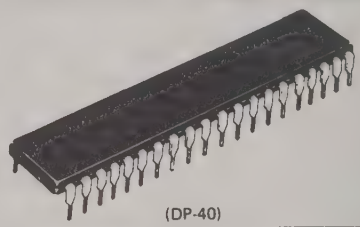
The HD46508 is a monolithic NMOS device with a 10-bit analog-to-digital converter, a programmable voltage comparator, a 16-channel analog multiplexer and HMCS6800 microprocessor family compatible interface.

Each of 16 analog inputs is either converted to a digital data by the analog-to-digital converter or compared with the specified value by the programmable comparator. The analog-to-digital converter uses successive approximation method as the conversion technique. It's intrinsic resolution is 10 bits but it can be 8 bits if the programmer so desires. The programmable voltage comparator compares the input voltage with the value specified by the programmer. The result (greater than, or smaller than) is reflected to the flag in the status register.

The device can expand its capability by controlling the external circuits such as sample holder, pre-amplifier and external multiplexer.

With these features, this device is ideally suited to applications such as process control, machine control and vehicle control.

HD46508P, HD46508P-1, HD46508PA, HD46508PA-1

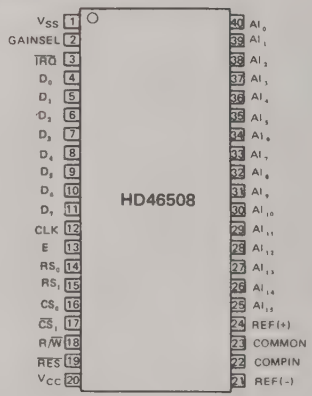


(DP-40)

■ FEATURES

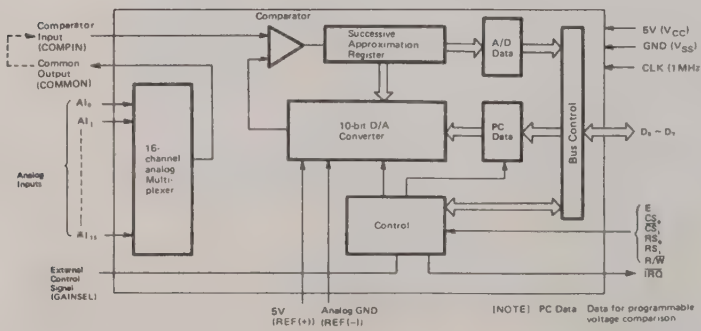
- 16-channel Analog multiplexer
- Programmable A/D Converter resolution (10-bit or 8-bit)
- Programmable Voltage comparison (PC)
- Conversion Time 100μs (A/D), 13μs(PC)
- External Sample and Hold Circuit Control
- Auto Range-switching Control of External Amplifier
- Waiting Function for the Settling Time of External Amplifier
- Interrupt Control (Only for A/D conversion)
- Single +5V Power Supply
- Compatible with HMCS6800 Bus (The connection with other Asynchronous Buses possible)

■ PIN ARRANGEMENT



(Top View)

■ BLOCK DIAGRAM



■ ORDERING INFORMATION

ADU	Bus Timing	Non Linearity*
HD46508A	1 MHz	±1 LSB
HD46508A-1	1.5 MHz	
HD46508	1 MHz	±3 LSB
HD46508-1	1.5 MHz	

* Specification for 10 bit A/D conversion

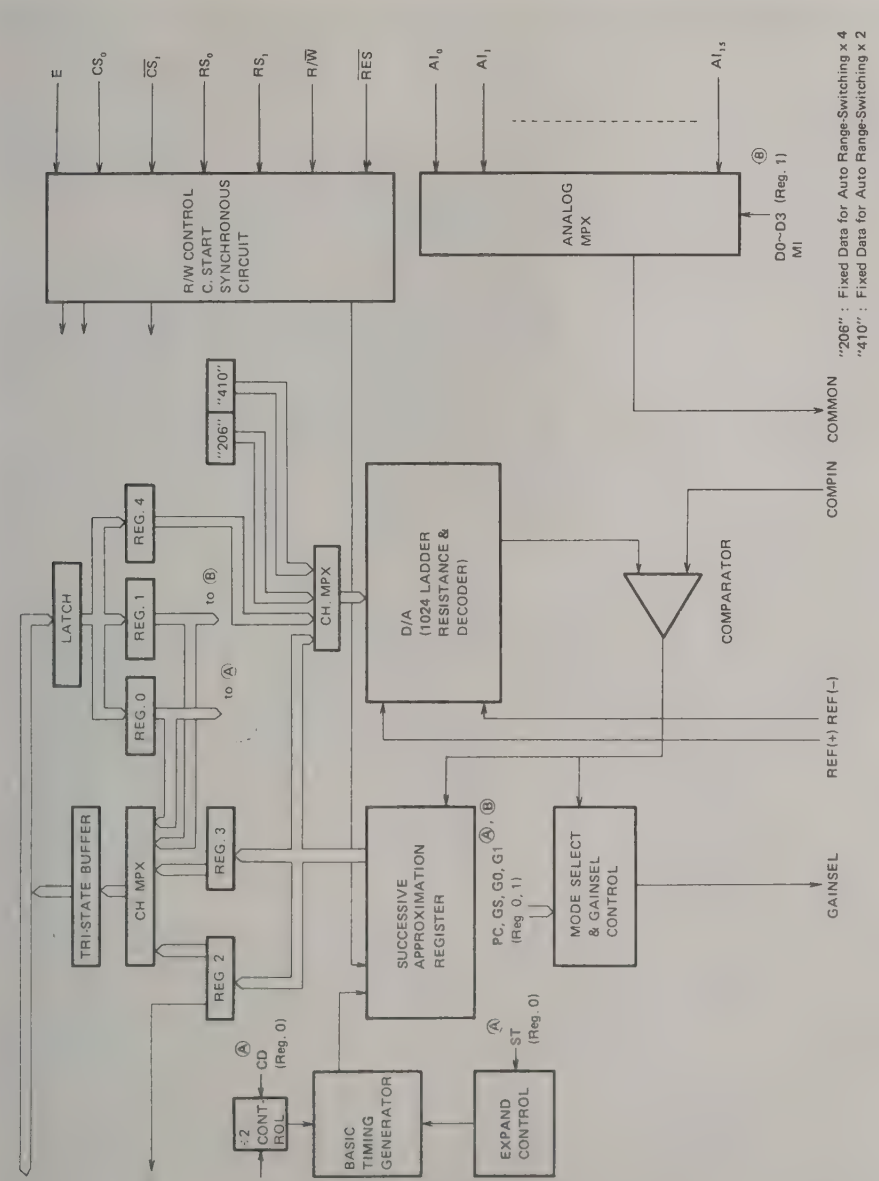


Figure 1 Internal Block Diagram

■ ABSOLUTE MAXIMUM RATINGS

Item	Symbol	Value	Unit
Supply Voltage	V_{CC}*	−0.3 ~ +7.0	V
Input Voltage	V_{in}*	−0.3 ~ +7.0	V
Analog Input Voltage	V_{Ain}*	−0.3 ~ +7.0	V
Operating Temperature	T_{opr}	− 20 ~ + 75	°C
Storage Temperature	T_{stg}	− 55 ~ +150	°C

* With respect to V_{SS} (SYSTEM GND)

[NOTE] Permanent LSI damage may occur if maximum ratings are exceeded. Normal operation should be under recommended operating conditions. If these conditions are exceeded, it could affect reliability of LSI.

■ RECOMMENDED OPERATING CONDITIONS

Item	Symbol	min	typ	max	Unit
Supply Voltage	V_{CC}*	4.75	5.0	5.25	V
Input "High" Voltage	V_{IH}*	2.0	–	V_{CC}	V
Input "Low" Voltage	V_{IL}*	−0.3	–	0.8	V
Analog Input Voltage	V_{Ain}*	0	–	$V_{REF(+)}$	V
Reference Voltage	$V_{REF(+)}$*	–	V_{CC}	V_{CC}+0.25	V
	$V_{REF(-)}$*	−0.1	0	–	
Voltage Center of Ladder	$\frac{V_{REF(+)} + V_{REF(-)}}{2}$*	–	$\frac{V_{CC}}{2}$	$\frac{V_{CC}}{2}$+0.25	V
Operating Temperature	T_{opr}	− 20	25	75	°C

*With respect to V_{SS} (SYSTEM GND)

■ ELECTRICAL CHARACTERISTICS

● DC CHARACTERISTICS <1> (V_{CC} = 5V ± 5%, V_{SS} = 0V, Ta = −20 ~ +75°C, unless otherwise noted.)

Item		Symbol	Test Condition	min	typ	max	Unit
Input "High" Voltage		V_{IH}		2.0	–	V_{CC}	V
Input "Low" Voltage		V_{IL}		−0.3	–	0.8	V
Output "High" Voltage	$D_0 \sim D_7$	V_{OH}	I_{OH} = −205μA	2.4	–	–	V
	GAINSEL		I_{OH} = −200μA	2.4	–	–	
			I_{OH} = −10μA	V_{CC}−1.0	–	–	
Output "Low" Voltage	$D_0 \sim D_7$, GAINSEL	V_{OL}	I_{OL} = 1.6 mA	–	–	0.4	V
	$\overline{IRQ}$		I_{OL} = 3.2 mA	–	–	0.4	
Input Leakage Current	E, CLK, R/$\overline{W}$ $\overline{RES}$, RS_0, RS_1 CS_0, $\overline{CS_1}$	I_{in}	V_{in} = 0 ~ 5.25V	-2.5	–	2.5	μA
Three-State (off state) Input Current	$D_0 \sim D_7$	I_{TSI}	V_{in} = 0.4 ~ 2.4V	-10	–	10	μA
Output Leakage Current	$\overline{IRQ}$	I_{LOH}	V_{OH} = 2.4V	–	–	10	μA
Power Dissipation		P_D		–	–	500	mW
Input Capacitance	$D_0 \sim D_7$	C_{in}	V_{in} = 0V, Ta = 25°C f = 1 MHz	–	–	12.5	pF
	E, CLK, R/$\overline{W}$ $\overline{RES}$, RS_0, RS_1 CS_0, $\overline{CS_1}$			–	–	10.0	pF
Output Capacitance	$\overline{IRQ}$, GAINSEL	C_{out}	V_{in} = 0V, Ta = 25°C f = 1 MHz	–	–	10.0	pF

- **DC CHARACTERISTICS <2>** (V_{CC} = 5V ± 5%, V_{SS} = 0V, Ta = −20 ~ +75°C, unless otherwise noted.)

Item	Test Condition	min	typ	max	Unit
Analog Multiplexer ON Resistance	V_{Ain} = 5.0V, V_{CC} = 4.75V, Ta = 25°C	—	—	1	kΩ
OFF Channel Leakage Current	V_{Ain} = 5.0V, V_{CC} = 4.75V, Ta = 25°C, COMMON = 0V	—	10	100	nA
	V_{Ain} = 0V, Ta = 25°C, V_{CC} = 4.75V, COMMON = 5V	−100	−10	—	nA
Analog Multiplexer Input Capacitance		—	—	7.5	pF
Ladder Resistance (from REF(+) to REF(−))	$V_{REF\ (+)}$ = 5.0V, $V_{REF\ (-)}$ = 0V, Ta = 25°C	10	—	40	kΩ

- **CONVERTER SECTION** (Ta = 25°C, V_{CC} = $V_{REF(+)}$ = 5.0V, t_{cycC} = 1μs, unless otherwise noted.)

1. 10-BIT A/D CONVERSION

Item	HD46508A, HD46508A-1			HD46508, HD46508-1			Unit
	min	typ	max	min	typ	max	
Resolution	—	10	—	—	10	—	bits
Non-linearity Error *	—	±1/2	±1	—	±1	±3	LSB
Zero-Error	—	±1/2	±3/4	—	±1/2	±1	LSB
Full-Scall Error	—	±1/4	±1/2	—	±1/2	±1	LSB
Quantization Error	—	—	±1/2	—	—	±1/2	LSB
Absolute Accuracy *	—	±1	±3/2	—	±2	±4	LSB

2. 8-BIT A/D CONVERSION

Item	HD46508A, HD46508A-1			HD46508, HD46508-1			Unit
	min	typ	max	min	typ	max	
Resolution	—	8	—	—	8	—	bits
Non-linearity Error *	—	±1/8	±1/4	—	±1/4	±3/4	LSB
Zero-Error	—	±1/4	±3/8	—	±3/8	±1/2	LSB
Full-Scall Error	—	±1/4	±3/8	—	±3/8	±1/2	LSB
Quantization Error	—	—	±1/2	—	—	±1/2	LSB
Absolute Accuracy *	—	±5/8	±3/4	—	±3/4	±5/4	LSB

3. PROGRAMMABLE VOLTAGE COMPARISON (PC)

Item	HD46508A, HD46508A-1			HD46508, HD46508-1			Unit
	min	typ	max	min	typ	max	
Resolution	—	8	—	—	8	—	bits
Non-linearity Error *	—	±1/8	±1/4	—	±1/4	±3/4	LSB
Zero-Error	—	±1/4	±3/8	—	±3/8	±1/2	LSB
Full-Scall Error	—	±1/4	±3/8	—	±3/8	±1/2	LSB
Absolute Accuracy *	—	±3/8	±5/8	—	±1/2	±1	LSB

*Temperature Coefficient; 25 ppm of FSR/°C (max)

- **AC CHARACTERISTICS (V_{CC} = 5.0V ± 5%, V_{SS} = 0V, Ta = −20 ~ +75°C, unless otherwise noted.)**

1. CLOCK WAVEFORM

Item	Symbol	Test Conditions	CD* = 0			CD* = 1			Unit
			min	typ	max	min	typ	max	
CLK Cycle Time	t_{cycC}	Fig. 2	1.0	–	10	0.5	–	5	μs
CLK "High" Pulse Width	PW_{CH}		0.45	–	4.5	0.22	–	2.2	μs
CLK "Low" Pulse Width	PW_{CL}		0.40	–	4.0	0.21	–	2.1	μs
Rise and Fall Time of CLK	t_{Cr}, t_{Cf}		–	–	25	–	–	25	ns

* CD : CLK Divider bit

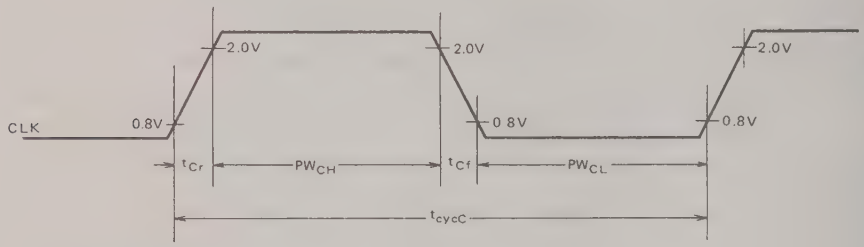


Figure 2 CLK Waveform

2. $\overline{IRQ}$, GAINSEL OUTPUT

Item	Symbol	Test condition	min	typ	max	Unit
$\overline{IRQ}$ Release Time	t_{IR}	Fig. 3	–	–	750	ns
GAINSEL Delay Time	t_{GSD1}	Fig. 4	–	–	750	ns
	t_{GSD2}		–	–	750	ns

t_{GSD1} : TTL Load
t_{GSD2} : CMOS Load

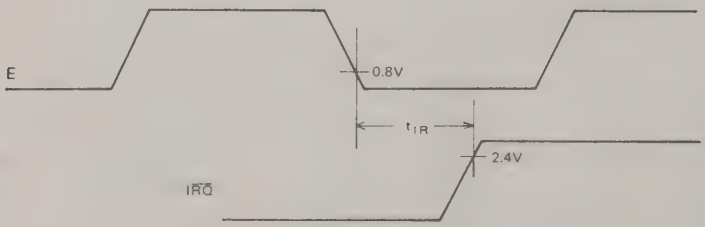
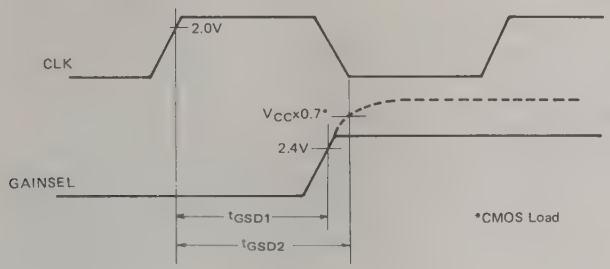


Figure 3 $\overline{IRQ}$ Release Time

(1) Sample & Hold



(2) x2, x4 Auto Range-Switching, Programmable Gain

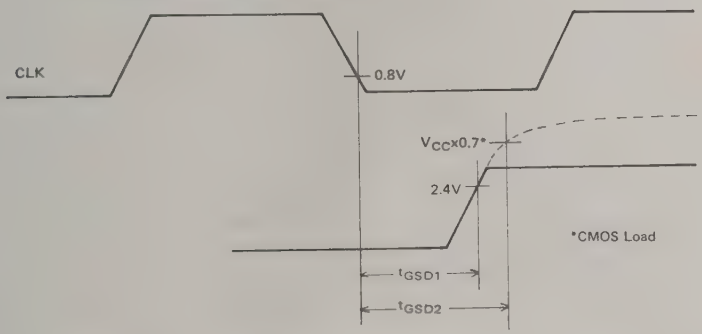


Figure 4 GAINSEL Delay Time

3. BUS TIMING CHARACTERISTICS

READ OPERATION SEQUENCE

Item	Symbol	Test Condition	HD46508 HD46508A			HD46508-1 HD46508A-1			Unit
			min	typ	max	min	typ	max	
Enable Cycle Time	t_{cycE}	Fig. 5	1.0	–	–	0.666	–	–	μs
Enable "High" Pulse Width	PW_{EH}		0.45	–	–	0.28	–	–	μs
Enable "Low" Pulse Width	PW_{EL}		0.40	–	–	0.28	–	–	μs
Rise and Fall Time of Enable	t_{Er}, t_{Ef}		–	–	25		–	25	ns
Address Set Up Time	t_{AS}		140	–	–	140	–	–	ns
Data Delay Time	t_{DDR}		–	–	320	–	–	220	ns
Data Access Time	t_{ACC}		–	–	460	–	–	360	ns
Data Hold Time	t_H		10	–	–	10	–	–	ns
Address Hold Time	t_{AH}		10	–	–	10	–	–	ns

WRITE OPERATION SEQUENCE

Item	Symbol	Test Condition	HD46508 HD46508A			HD46508-1 HD46508A-1			Unit
			min	typ	max	min	typ	max	
Enable Cycle Time	t_{cycE}	Fig. 6	1.0	–	–	0.666	–	–	μs
Enable "High" Pulse Width	PW_{EH}		0.45	–	–	0.280	–	–	μs
Enable "Low" Pulse Width	PW_{EL}		0.40	–	–	0.280	–	–	μs
Rise and Fall Time of Enable	t_{Er}, t_{Ef}		–	–	25	–	–	25	ns
Address Set Up Time	t_{AS}		140	–	–	140	–	–	ns
Data Set Up Time	t_{DSW}		195	–	–	80	–	–	ns
Data Hold Time	t_H		10	–	–	10	–	–	ns
Address Hold Time	t_{AH}		10	–	–	10	–	–	ns

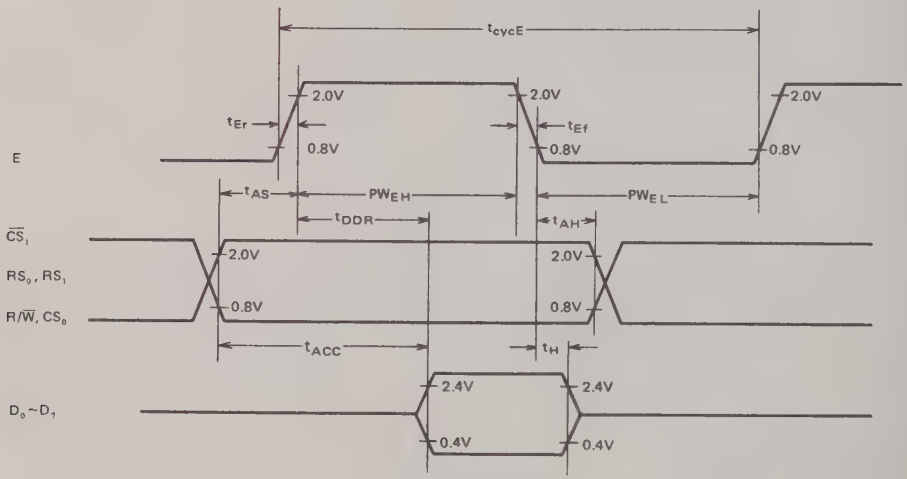


Figure 5 Read Timing

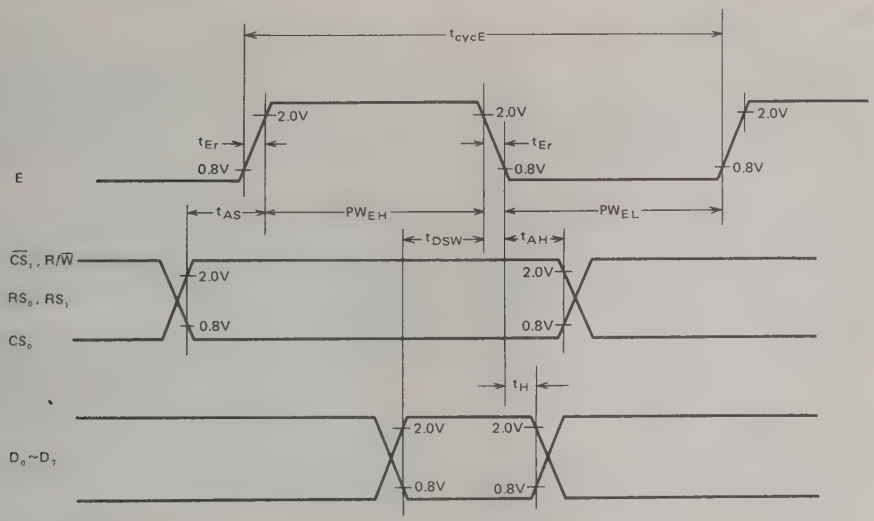


Figure 6 Write Timing

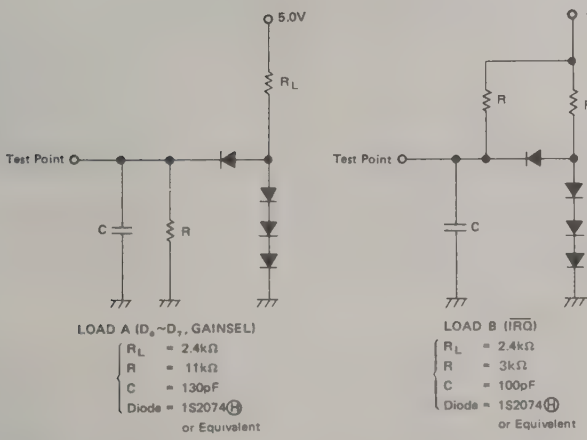


Figure 7 Test Load

■ SIGNAL DESCRIPTION

• Processor Interface

Data Bus (D_0~D_7)

The Bi-directional data lines (D_0~D_7) allow data transfer between the ADU and MPU. Data bus output drivers are three state buffers that remain in the high-impedance state except when MPU performs a ADU read operation.

Enable (E)

The Enable signal (E) is used as strobe signal in MPU R/W operation with the ADU internal registers. This signal is normally derived from the HMCS6800 system clock (ϕ_2).

Chip Select (CS_0, $\overline{CS_1}$)

The Chip Select lines (CS_0, $\overline{CS_1}$) are used to address the ADU. The ADU is selected when CS_0 is at "High" and $\overline{CS_1}$ is at "Low" level.

Read/Write (R/$\overline{W}$)

The R/$\overline{W}$ line controls the direction of data transfer between the ADU and MPU. When R/$\overline{W}$ is at "High" level, data of ADU is transferred to MPU. When R/$\overline{W}$ is at "Low" level, data of MPU is transferred to ADU.

Register Select (RS_0, RS_1)

The Register Select line (RS_0, RS_1) are used to select one of the 4 ADU internal registers. Table 1 shows the relation between (RS_0, RS_1) address and the selected register. The lowest 2 address lines of MPU are usually used for these signals.

Reset ($\overline{RES}$)

This input is used to reset the ADU. An input "Low" level on $\overline{RES}$ line forces the ADU into following status.

1) All the shift-registers in ADU are cleared and the conversion operation is stopped.
2) The GAINSEL output goes down to "Low" level. The $\overline{IRQ}$ output is made "Off" state and the D_0~D_7 are made high impedance state.

Interrupt Request ($\overline{IRQ}$) (Open Drain Output)

This output line is used to inform the A/D conversion end signal to the MPU. This signal becomes active "Low" level when IE bit in the control register 1 is "1" and IRQ bit in the control register 2 goes "1" at the end of conversion. And this signal returns to "High" right after The MPU reads the A/D Data Register (R3). Programmable voltage comparison does not affect this signal.

• Analog Data Interface

Analog Input (AI_0~AI_{15})

The Input Analog Data to be measured is applied to these Analog Input (AI_0~AI_{15}). These are multiplexed by internal 16 channel multiplexer and output to COMMOM pin. A particular input channel is selected when the multiplexer channel address is programmed into the control Register 1 (R1).

Multiplexer Common Output (COMMON)

This signal is the output of the 16 channel analog multiplexer, and may be connected to the input of pre-amplifier or sample/hold circuit according to user's purposes. When no external circuit needed, this output should be connected to the COMPIN input.

Comparator Input (COMPIN)

This is a high impedance input line that is used to transmit selected analog data to comparator. The COMMON line is usually connected to this input. When external Pre-amplifier or Sample/hold circuit is used, output of these circuits may be connected to this input.

Reference Voltage (+) (REF (+))

This line is used to apply the standard voltage to the internal ladder resistors.

Reference Voltage (–) (REF (–))

This line is connected to the analog ground.

• ADU Control

Conversion Clock (CLK)

The CLK is a standard clock input signals which defines internal timing for A/D conversion and PC operation.

Gain Select (GAINSEL) (CMOS Compatible Output)

This output is used to control the external circuit. The function of this signal is programmable and it is specified by (G1, G0) bits in Control Register 0. By using this output, user can control the auto-range-switching of external pre-amplifier, also control external sample & hold circuit, etc. as well.

[NOTE] This LSI is different from other HMCS6800 family LSIs in following function
- $\overline{RES}$ doesn't affect IE bit of R0

■ FUNCTION OF INTERNAL REGISTERS

• Structure

Table 1 Internal Registers of the ADU

$\overline{CS_1}$	CS_0	RS_1	RS_0	Reg. #	Register Name	Read	Write	Data Bit 7	6	5	4	3	2	1	0
0	1	0	0	R0	Control Register 0	○	○	IE	CD	ST				G1	G0
0	1	0	1	R1	Control Register 1	○	○	SC	GS	PC	MI	D3	D2	D1	D0
0	1	1	0	R2	Status & A/D Data Register (H)	○	×	IRQ	BSY	PCO		OV	DW	C9	C8
0	1	1	1	R3	A/D Data Register (L)	○	×	C7	C6	C5	C4	C3	C2	C1	C0
0	1	1	1	R4	PC Data Register	×	○	B7	B6	B5	B4	B3	B2	B1	B0

(Note) ○ --- YES
× --- NO

Control Register 0 (R0)

7	6	5	4	3	2	1	0
IE	CD	ST				G1	G0

	"1"	"0"
Mode Select	See Table 2	
Not Used		
Not Used		
Not Used		
Settling Time	Available	Not Available
CLK Divider	CLK/2	CLK
Interrupt Enable*	Enable IRQ	Mask IRQ

*$\overline{RES}$ doesn't affect IE bit.

Figure 8 Control Register 0

Control Register 1 (R1)

7	6	5	4	3	2	1	0
SC	GS	PC	MI	D3	D2	D1	D0

	"1"	"0"
MPX Channel Address	See Table 3	
MPX Inhibit	Inhibited	Not Inhibited
Prog. Comparator Select	Prog. Comparator mode	A/D Converter mode
GAINSEL Enable	GAINSEL Enable	GAINSEL Disable
Short-cycle Conversion	8-bit Length	10-bit Length

Figure 9 Control Register 1

Status & A/D Data Register (H)

7	6	5	4	3	2	1	0
IRQ	BSY	PCO		OV	DW	C9	C8

	"1"	"0"
Upper bit (10 bit data)		
Data Weight	See Table 4.	
Data Over Scale flag	Data is over scale	Within the scale
Not Used		
Programmable Comparator Output	$V_{Ain} > V_p$	$V_{Ain} < V_p$
Busy flag	Under Conversion	Conversion Completed
IRQ flag	Requested	Not Requested

V_{Ain} : Unknown Input Voltage
V_p : Programmed Voltage by R4
C9, C8 bits are cleared when 8 bit A/D conversion is performed.

Figure 10 Status & A/D Data Register (H)

A/D Data Register (L)

7	6	5	4	3	2	1	0
C7	C6	C5	C4	C3	C2	C1	C0

Lower order 8 bit Data (Normal 10 bit Conversion)
8 bit Data (8 bit Short-cycle Conversion)

Figure 11 A/D Data Register (L)

PC Data Register

7	6	5	4	3	2	1	0
B7	B6	B5	B4	B3	B2	B1	B0

8 bit Data for Programmable Voltage Comparison

Figure 12 PC Data Register

- **Description for the Internal Registers**

Control Register 0 (R0)

This Register is a 5-bit read/write register that is used to specify Interrupt Enable (IE), CLK Divider (CD), Settling Time (ST) and Mode Select (G0, G1). This Register should be written before writing R1.

IE bit: (Interrupt Enable)
- IE = "1", Interrupt is requested through the $\overline{IRQ}$ output.
- IE = "0", Interrupt is masked.

CD bit: (Clock Divider)
- CD = "1", CLK ÷ 2 is used as internal clock.
- CD = "0", CLK is used directly.

ST bit: (Settling Time)
- ST = "1", First comparison is executed after 1 expanded cycle in order to compensate external amplifiers settling delay.
- ST = "0", Cycle is not delayed.

G0, G1 bit; (Mode select) These bits are used to specify the function of GAINSEL signal when GS bit is "1".

Table 2 Function of G0, G1

G1	G0	Mode Select
0	0	Sample & Hold
0	1	Auto Range-Switching x 2
1	0	Auto Range-Switching x 4
1	1	Programmable Gain Control

Control Register 1 (R1)

This register is an 8-bit read/write register that is used to store the command for A/D conversion mode and programmable comparison mode. This register includes MPX channel address (D_0~D_3), MPX inhibit (MI), programmable comparator select (PC), GAINSEL enable (GS) and short-cycle conversion (SC) bits. When this register (R1) is programmed, each conversion mode starts.

SC bit (Short-cycle)
- SC = "1", Short-cycle conversion (8 bit length)
- SC = "0", Normal conversion (10 bit length)

GS bit (GAINSEL Enable)
- GS = "1", GAINSEL signal is enabled. The function of GAINSEL is specified by (G0, G1) bits.
- GS = "0", GAINSEL signal is disabled. ("Low" level)

PC bit (Program comparator)
- PC = "1", Programmable voltage comparator mode
- PC = "0", A/D conversion mode

MI bit (MPX Inhibit)
- MI = "1", Internal MPX channel is inhibited in order to use external MPX channel.
- MI = "0", Internal MPX channel is used.

D0~D3 (MPX channel) These bits are used to select the particular MPX channel.

Table 3 MPX Channel Addressing

Channel #1	D3	D2	D1	D0	Analog Input
0	0	0	0	0	AI_0
1	0	0	0	1	AI_1
2	0	0	1	0	AI_2
3	0	0	1	1	AI_3
4	0	1	0	0	AI_4
5	0	1	0	1	AI_5
6	0	1	1	0	AI_6
7	0	1	1	1	AI_7
8	1	0	0	0	AI_8
9	1	0	0	1	AI_9
10	1	0	1	0	AI_{10}
11	1	0	1	1	AI_{11}
12	1	1	0	0	AI_{12}
13	1	1	0	1	AI_{13}
14	1	1	1	0	AI_{14}
15	1	1	1	1	AI_{15}

Table 4 Function Select

PC	SC	Function	GS	(G0, G1)
0	0	10 bit AD CONV.	0	DISABLE
			1	ENABLE*
	1	8 bit AD CONV.	0	DISABLE
			1	ENABLE*
1	x	PROG. COMP (8 bit)	x	DISABLE

x = Do not care
* = See Table 6
[NOTE] CD bit and ST bit are effective in every case.

Status & A/D Data Register (H) (R2)

This register is a 7-bit read only register that is used to store the upper 2-bit data (C8, C9), data weight (DW), data overscale (OV), programmable comparator output (PCO), busy (BSY) and interrupt request(IRQ).

(C8, C9) (Upper bit data) : These bits store upper 2-bit data measured by 10 bit length conversion.

DW bit (Data weight) : This bit indicates data weight when Auto range-switching mode is selected. This bit is set or reset when the conversion has completed. The conditions are shown in following Table.
In this mode GAINSEL output also goes "High" or "Low" on the same condition shown in Table 5.
Other status of DW bit is shown in Table 6.

OV bit (Over scale) : This bit is set when analog data is greater than or equal to reference Voltage ($V_{REF(+)}$).

PCO bit (Programmable comparator Output) : This bit indicates the result of programmable voltage comparison.
"1" → PCO $V_{Ain} > Vp$
"0" → PCO $V_{Ain} < Vp$
V_{Ain} : Analog Input Voltage to be compared
Vp : Programmed Voltage (R4)

BSY bit (Busy) : This bit indicates that the ADU is now under conversion.

IRQ bit (Interrupt Request) : This bit is set when the A/D conversion has completed and cleared by reading the R3.

A/D Data Register (L) (R3)

This register is an 8-bit read-only register that is used to store the lower 8 bits data of 10-bit conversion or full 8 bits data of the 8-bit conversion.

PC Data Register (R4)

This register is an 8-bit write-only register prepared for Programmable Voltage comparison. Stored data is converted to digital voltage, and compared with analog input to be measured. The result of comparison is set into PCO bit.

Table 5 Data Weight (DW) Set or Reset Condition

Mode \ Condition	Set ("1")	Reset ("0")
Auto Range-Switching (x2)	$V_{Ain} < \frac{410}{1024} \cdot V_{REF(+)}$	$V_{Ain} > \frac{410}{1024} \cdot V_{REF(+)}$
Auto Range-Switching (x4)	$V_{Ain} < \frac{206}{1024} \cdot V_{REF(+)}$	$V_{Ain} > \frac{206}{1024} \cdot V_{REF(+)}$

V_{Ain} : Analog Input Voltage to be measured
$V_{REF(+)}$: Voltage Applied to REF(+)

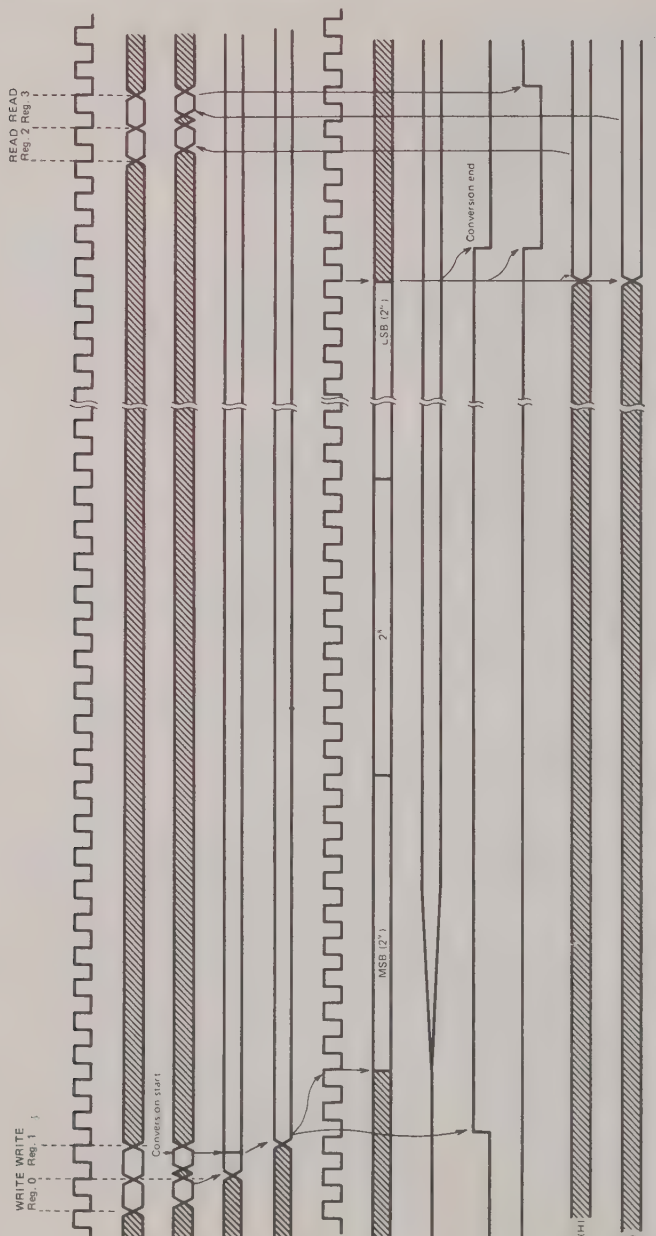
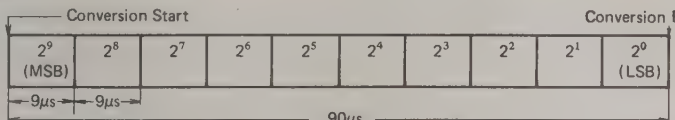


Figure 13 A/D Conversion Timing Chart (Basic Sequence)

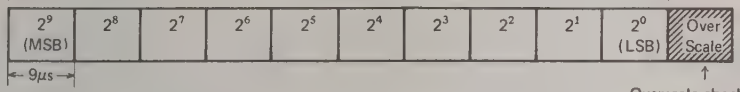
- **A/D Conversion and PC sequence** (t_{cyc}=1μs)

10 bits A/D Conversion

1) Basic Sequence
(SC = "0", ST = "0", GS = "0")

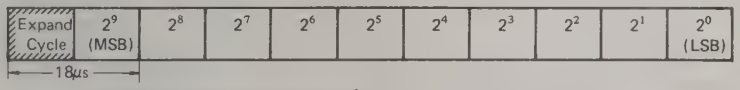


2) Basic Sequence (When overscale is detected)



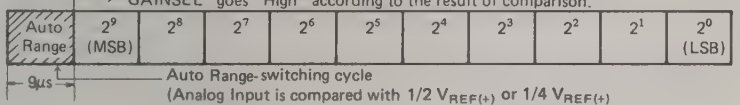
Overscale check Cycle (Analog Input is compared with $V_{REF(+)}$.)

3) Expanded Sequence
(SC = "0", ST = "1", GS = "0")



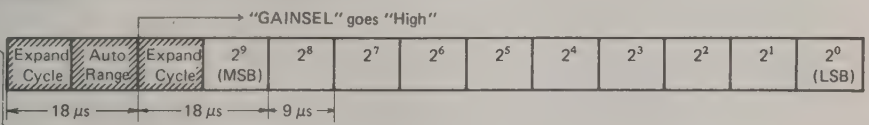
MSB cycle is expanded to compensate external amplifier's settling delay.

4) Auto Range-Switching Control Sequence
(SC = "0", ST = "0", GS = "1", G0 = "0", G1 = "1") or (G0 = "1", G1 = "0")

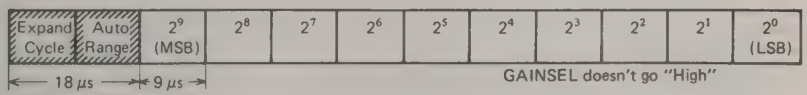


5) Auto Range-Switching & Expansion Control Sequence
(SC = "0", ST = "1", GS = "1", G0 = "0", G1 = "1") or (G0 = "1", G1 = "0")

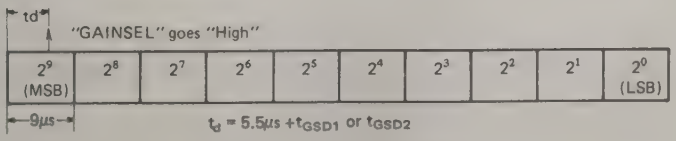
a) Analog Input < 1/2 $V_{REF(+)}$ or 1/4 $V_{REF(+)}$



b) Analog Input > 1/2$V_{REF(+)}$ or 1/4$V_{REF(+)}$

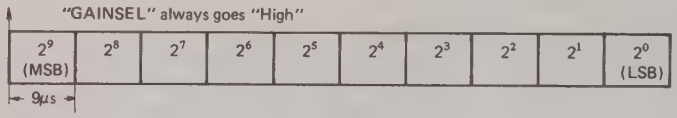


6) Sample & Hold Control Sequence
(SC = "0", ST = "0", GS = "1", G0 = "0", G1 = "0")

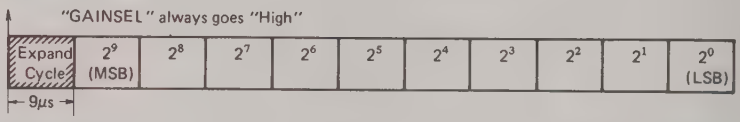


t_d = 5.5μs +t_{GSD1} or t_{GSD2}

7) Programmable Gain Control Sequence
(SC = "0", ST = "0", GS = "1", G0 = "1", G1 = "1")

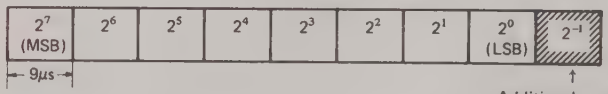


8) Programmable Gain & Expansion Control Sequence
(SC = "0", ST = "1", GS = "1", G0 = "1", G1 = "1")



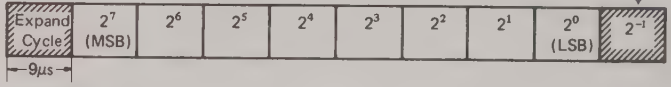
8 Bit A/D Conversion

1) Basic Sequence
(SC = "1", ST = "0", GS = "0")



Additional conversion cycle for rounding the LSB – 1 Bit.

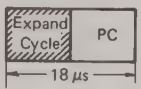
2) Expanded Sequence
(SC = "1", ST = "1", GS = "0")



Programmable Voltage Comparison

1) Basic Sequence
(PC = "1", ST = "0")

2) Expanded Sequence
(PC = "1", ST = "1")



■ HOW TO USE THE ADU

• Functions of GAINSEL

The ADU is equipped with programmable GAINSEL output signal. By using GAINSEL output and external circuit, the ADU is able to implement following control.

1) Auto Range-Switching (Auto Gain) Control
2) Programmable Gain control
3) Sample & Hold control

GAINSEL output is controlled by Mode Select bit (G0, G1) when GAINSEL enable bit (GS) is "1".

Table 6 GAINSEL Control

GS	G1	G0	GAINSEL	Control Mode	DW
0	x	x	"Low"	Normal Use (GAINSEL is not used)	0
1	0	0	"High"	Sample & Hold control	0
1	0	1	*	Auto Range Switching x 2 control	**
1	1	0	*	Auto Range Switching x 4 control	**
1	1	1	"High"	Programmable Gain control	1

* GAINSEL goes "High" or "Low" according to the condition shown in Table 5.
** See, Table 5.

How to Control External Circuit

(1) Sample & Hold Control (G1=0, G0=0)

An example of Sample & Hold circuit is shown in Fig. 14. When ADU is set in Sample & Hold Control Mode, GAINSEL becomes "High" level on conversion and controls the data holding.

(2) Automatic Range Switching Control (G1=0, G0=1 or G1=1, G0=0)

The GAINSEL signal controls the external amplifier which can change the ratio of voltage amplification. (GAIN: 1 → 2 times or 1 → 4 times). Fig. 15 shows Automatic Range Switching Control. In this case, when the input voltage is lower than 206/1024 $V_{REF(+)}$, GAINSEL becomes "High" level. This makes the GAIN of the amplifier change from 1 to 4 times, and 4 times value of the input voltage is A/D converted. Using this function even if an input signal is small, it is possible to execute A/D conversion in nearly full scale. In this mode, when GAINSEL signal becomes "High", DW bit becomes "1" to show the range switching is in a progress.

(3) Programmable GAIN Control (G1=1, G0=1)

The GAINSEL signal is used for controlling the external amplifier of any GAIN which is fit to the system.

In this mode, GAINSEL always becomes "High" at the beginning of A/D conversion, so the change of range is controlled by GS bit. Converted data need to be corrected in software in accordance with GAIN of the amplifier.

This mode is effective in the case of converting very small input voltage.

(Note) Refer to "ADU Function Sequence" (A/D Conversion and PC Sequence) for the timing in which GAINSEL signal becomes "High". GAINSEL signal becomes "Low" in accordance with "1" → "0" change of BSY bit. Refer to Fig. 13.

x1 Sample & Hold

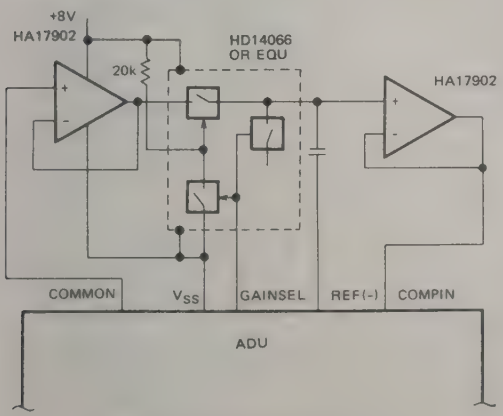


Figure 14 Sample & Hold Circuit

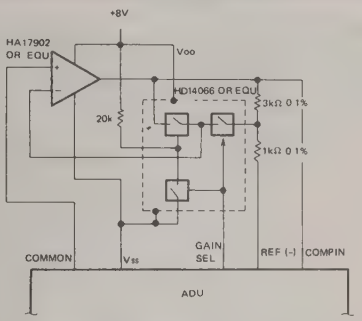
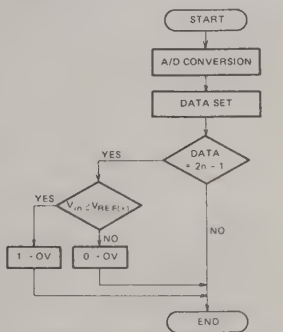


Figure 15 Pre-amplifier Circuit
(x1, x4 Auto-Range Switching)

- **Overscale Check**

ADU is equipped with hardware overscale detection function. The overscale detection is performed automatically when the result of A/D conversion is 2^n-1 (all bits = 1). When analog input V_{Ain} is higher than $V_{REF(+)}$, overscale bit (OV) is set to "1". The definition of the overscale is illustrated in Fig. 17. And the flow of overscale check is shown in Fig. 16.



OV	DATA	NOTE
0	11 ··· 1	NOT OVERSCALE
1	11 ··· 1	OVERSCALE

Figure 16 Overscale Check Flow

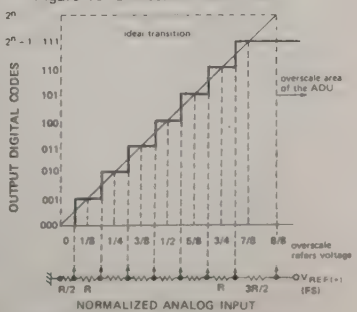


Figure 17 Definition ADU's Overscale

- **Usage of the PC**

The ADU has a programmable threshold voltage comparator (PC) function. The threshold voltage is pre-setable from 0V to 5V range with 8 bit resolution. The comparator's output is stored into PCO bit at the end of comarison.

The programmable voltage comparison time is so short that the interrupt is not requested at this mode. The end of comparison needs to be confirmed by reading the 1→0 transition of the BSY bit in R2.

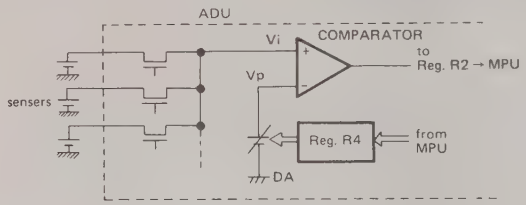
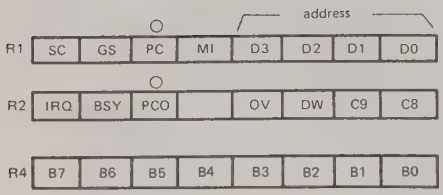


Figure 18 Function Diagram of the PC



PC=0 : A/D conversion mode
PC=1 : Programmable Voltage Comparison Mode

PCO : Programmable comparator output (1 bit data)

B_0~B_7 : V_P setting byte (upper byte of 10 bit D/A. Lower byte is set to 0)

Figure 19 Registers of the PC Mode

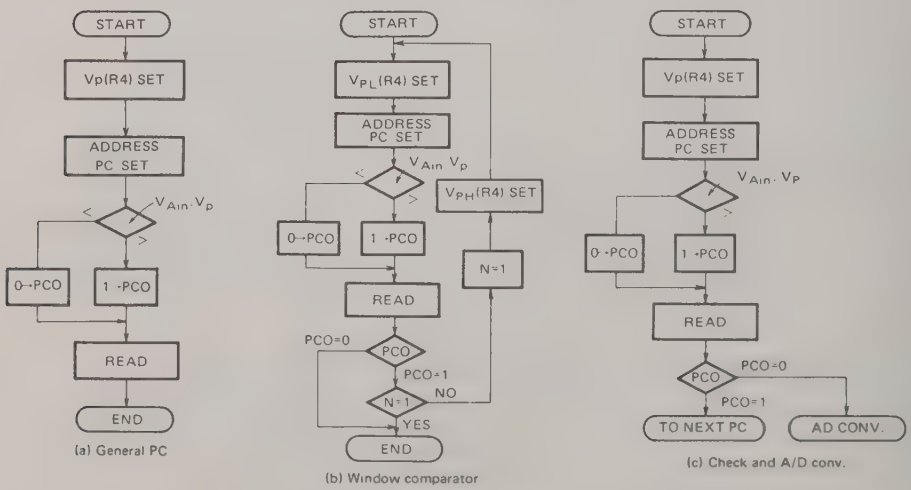
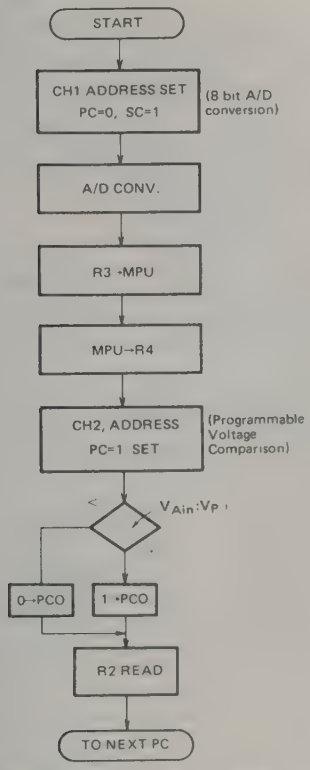


Figure 20 PC Application Flow Chart Examples



(d) Voltage Comparison between two channels.

Figure 20 PC Application Flow Chart Examples (continued)

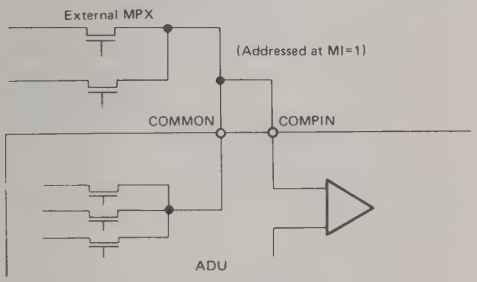
- **How to use MI bit**

MI bit (R1) functions as follows.

MI = 1: Internal MPX channel is inhibited in order to use attached external MPX channel.

MI = 0: Internal MPX channel is enabled.

MI bit used to select either of External MPX and Internal MPX. External MPX is connected as follows.



[NOTE] When external MPX is used as the way figure 20, 1 dammy AD conversion or PC at MI=1 should be performed.

Figure 21 How to use External MPX

- **EXAMPLE OF APPLIED CIRCUIT OF THE ADU**

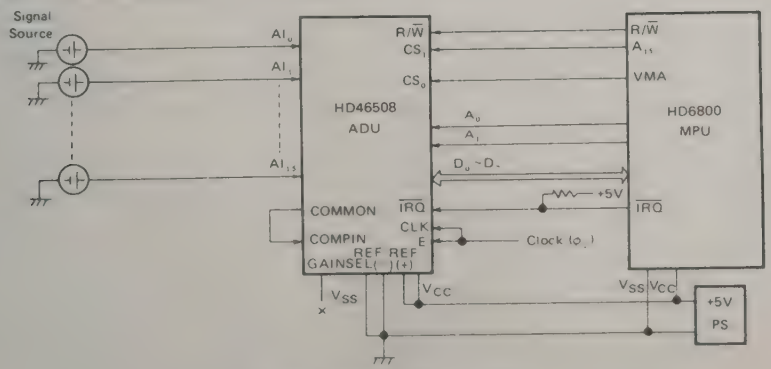


Figure 22 Single ADU System

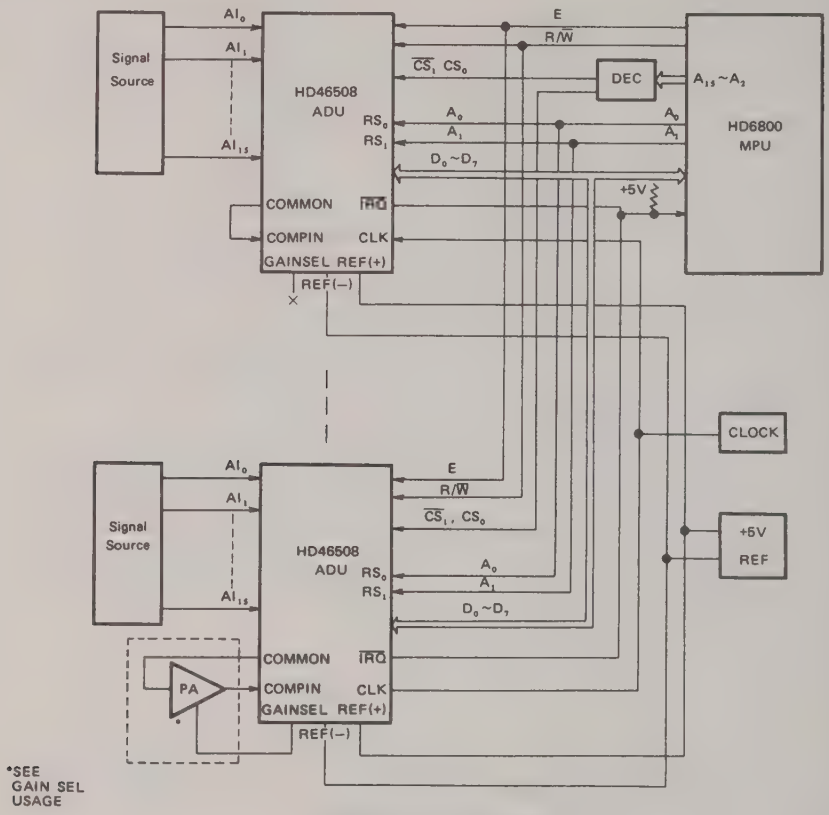


Figure 23 Multi ADU System

■ DEFINITIONS OF ACCURACY

Definitions of accuracy applyed to HD46508 are as follows.

(1) Resolution . . . The number of output binary digit.

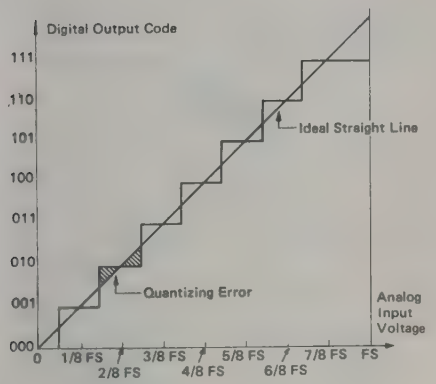
(2) Offset Error . . . The difference between actual input voltage and ideal input voltage for the first transition. (when digital output code is changed from 000 . . . 00 to 000 . . . 01.)

(3) Full Scale Error . . . The difference between actual input voltage and ideal input voltage for the final transition. (when digital output code is changed from 111 . . . 10 to 111 . . . 11.)

(4) Quantizing Error . . . Error equipped in A/D converter inherently. Always ±½ LSB is applied.

(5) Non-linearity Error . . . The maximum deviation of the actual transfer line from an ideal straight line. This error doesn't include Quantizing Error, Offset, or Full Scale Errors.

(6) Absolute Accuracy . . . The deviation of the digital output code from an analog input voltage. Absolute accuracy includes all of (2), (3), (4), (5).



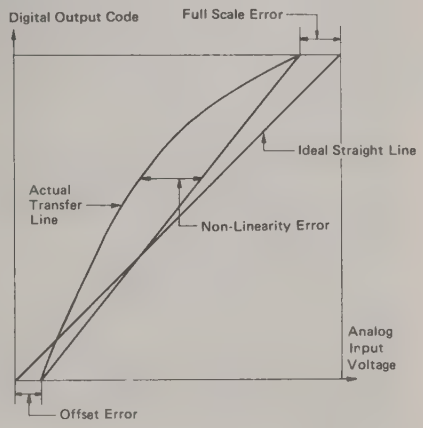


Figure 24 Definition of Accuracy

HD146818

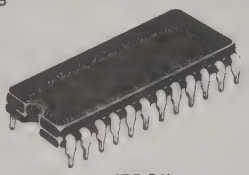
RTC (Real Time Clock Plus RAM)

The HD146818 is a HMCS6800 peripheral CMOS device which combines three unique features: a complete time-of-day clock with alarm and one hundred calendar, a programmable periodic interrupt and square-wave generator, and 50 bytes of Low-power static RAM.

This device includes HD6801, HD6301 multiplexed bus interface circuit and 8085's multiplexed bus interface as well, so it can be directly connected to HD6801, HD6301 and 8085.

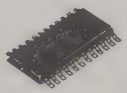
The Real-Time Clock plus RAM has two distinct uses. First, it is designed as battery powered CMOS part including all the common battery backed-up functions such as RAM, time, and calender. Secondly, the HD146818 may be used with a CMOS microprocessor to relieve the software of timekeeping workload and to extend the available RAM of an MPU such as the HD6301.

HD146818P



(DP-24)

HD146818FP



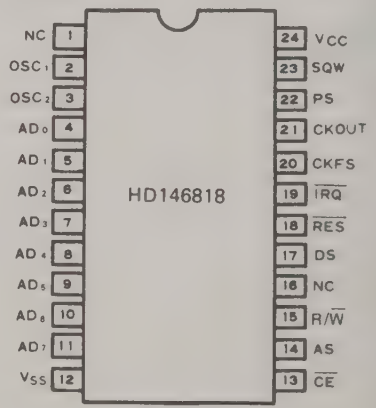
(FP-24)

The Flat Package product is under development.

■ FEATURES

- Time-of-Day Clock and Calendar
  - Counts Seconds, Minutes, and Hours of the Day
  - Counts Days of Week, Date, Month, and Year
- Binary or BCD Representation of Time, Calendar, and Alarm
- 12- or 24 Hour Clock with AM and PM in 12-Hour Mode
- Automatic End of Month Recognition
- Automatic Leap Year Compensation
- Interfaced with Software as 64 RAM Locations
  - 14 Bytes of Clock and Control Register
  - 50 Bytes of General Purpose RAM
- Three Interrupt are Separately Software Maskable and Testable
  - Time-of-Day Alarm, Once-per-Second to Once-per-Day
  - Periodic Rates from 30.5μs to 500ms
  - End-of-Clock Update Cycle
- Programmable Square-Wave Output Signal
- Three Time Base Input Options
  - 4.194304 MHz
  - 1.048576 MHz
  - 32.768 kHz
- Clock Output May be used as Microprocessor Clock Input
  - At Time Base Frequency ÷4 or ÷1
- Multiplexed Bus Interface Circuit of HD6801, HD6301 and 8085
- Low-Power, High-Speed, High-Density CMOS
- Battery Backed-up Operation
- Motorola MC146818 Compatible

■ PIN ARRANGEMENT



(Top View)

■ ABSOLUTE MAXIMUM RATINGS

Item	Symbol	Value	Unit
Supply Voltage	V_{CC} *	-0.3 ~ +7.0	V
Input Voltage	V_{in} *	-0.3 ~ +7.0	V
Operating Temperature	T_{opr}	0 ~ +70	°C
Storage Temperature	T_{stg}	-55 ~ +150	°C

* With respect to V_{SS} (SYSTEM GND)

(NOTE) Permanent LSI damage may occur if maximum rating are exceeded. Normal operation should be under recomended operating condition. If these conditions are exceeded, it could affect reliability of LSI.

■ RECOMENDED OPERATING CONDITIONS

Item	Symbol	min	typ	max	Unit
Supply Voltage	V_{CC} *	4.5	5.0	5.25	V
Input Voltage	V_{IL} *	-0.3	–	0.7	V
	V_{IH} *	V_{CC}-1.0	–	V_{CC}	V
Operating Temperature	T_{opr}	0	25	70	°C

* With respect to V_{SS} (SYSTEM GND)
(NOTE) Refer to Battery Backed-up Electrical characteristics.

■ ELECTRICAL CHARACTERISTICS

● DC CHARACTERISTICS (V_{CC} = 4.5 ~ 5.25V, V_{SS} = 0V, Ta = 0 ~ +70°C, unless otherwise noted.)

Item		Symbol	Test Condition	min	typ	max	Unit
Input "High" Voltage	$AD_0 \sim AD_7$, $\overline{CE}$, AS, R/$\overline{W}$, DS, CKFS, PS	V_{IH}		V_{CC}-2.0	–	V_{CC}	V
	$\overline{RES}$			V_{CC}-1.0	–	V_{CC}	
	OSC_1			V_{CC}-1.0	–	V_{CC}	
Input "Low" Voltage	$AD_0 \sim AD_7$, $\overline{CE}$, AS, R/$\overline{W}$, DS, CKFS, PS	V_{IL}		-0.3	–	0.7	V
	$\overline{RES}$			-0.3	–	0.8	
	OSC_1			-0.3	–	0.8	
Input Leakage Current	OSC_1, $\overline{CE}$, AS, R/$\overline{W}$, DS, $\overline{RES}$, CKFS, PS	$\|I_{in}\|$		–	–	2.5	μA
Three-state (off state) Input Current	$AD_0 \sim AD_7$	$\|I_{TSI}\|$		–	–	10	μA
Output Leakage Current	$\overline{IRQ}$	I_{LOH}		–	–	10	μA
Output "High" Voltage	$AD_0 \sim AD_7$	V_{OH}	I_{OH} = -1.6 mA	4.1	–	–	V
	SQW, CKOUT						
	$AD_0 \sim AD_7$		I_{OH} < -10 μA	V_{CC}-0.1	–	–	V
	SQW, CKOUT						
Output "Low" Voltage	$AD_0 \sim AD_7$	V_{OL}	I_{OL} = 1.6 mA	–	–	0.5	V
	CKOUT		I_{OL} = 1.6 mA				
	$\overline{IRQ}$, SQW		I_{OL} = 1.6 mA				
Input Capacitance	$AD_0 \sim AD_7$	C_{in}	V_{in} = 0V, Ta = 25°C, f = 1 MHz	–	–	12.5	pF
	All inputs except $AD_0 \sim AD_7$			–	–	12.5	pF
Output Capacitance	SQW, CKOUT, $\overline{IRQ}$	C_{out}		–	–	12.5	pF
Supply Current (MPU Read/Write operating) — Crystal Oscillation	f_{OSC} = 4 MHz	I_{CC} *	V_{CC} = 5.0V, SQW: disable, CKOUT = fosc (No Load), t_{cyc} = 1 μs, Circuit: Fig. 11, Parameter: Table 1	–	–	10	mA
	f_{OSC} = 1 MHz			–	–	7	
	f_{OSC} = 32 kHz			–	–	5	
Supply Current ** (MPU not operating) — Crystal Oscillation	f_{OSC} = 4 MHz			–	–	5	mA
	f_{OSC} = 1 MHz			–	–	2	
	f_{OSC} = 32 kHz			–	300	500	μA
Supply Current (MPU Read/Write operating) — External Clock	f_{OSC} = 4 MHz	I_{CC} *	V_{CC} = 5.0V, SQW: disable, CKOUT = fosc (No Load), OSC_2: open, t_{cyc} = 1 μs, Circuit: Fig. 17	–	–	10	mA
	f_{OSC} = 1 MHz			–	–	7	
	f_{OSC} = 32 kHz			–	–	5	
Supply Current ** (MPU not operating) — External Clock	f_{OSC} = 4 MHz			–	–	4	mA
	f_{OSC} = 1 MHz			–	–	1	
	f_{OSC} = 32 kHz			–	60	100	μA

* Supply current of HD146818 is defined as the value when the time-base frequency to be used is programmed into Register A. When power is turned on, these bits are unfixed, so there is a case that current more than the above specification may flow. Please never fail to set the time-base frequency after turning on power supply.

** $V_{IH\ min} = V_{CC}$-0.2V
$V_{IL\ max} = V_{SS}$+0.2V

- **AC CHARACTERISTICS (V_{CC} = 4.5 ~ 5.25V, V_{SS} = 0V, Ta = 0 ~ +70°C, unless otherwise noted.)**

BUS TIMING

Item	Symbol	min	typ	max	Unit
Cycle Time	t_{cyc}	953	–	–	ns
Pulse Width, AS/ALE "High"	PW_{ASH}	100	–	–	ns
AS Rise Time	t_{ASr}	–	–	30	ns
AS Fall Time	t_{ASf}	–	–	30	ns
Delay Time DS/E to AS/ALE Rise	t_{ASD}	40	–	–	ns
DS Rise Time	t_{DSr}	–	–	30	ns
DS Fall Time	t_{DSf}	–	–	30	ns
Pulse Width, DS/E Low or $\overline{RD}/\overline{WR}$ "High"	PW_{DSH}	325	–	–	ns
Pulse Width, DS/E High or $\overline{RD}/\overline{WR}$ "Low"	PW_{DSL}	300	–	–	ns
Delay Time, AS/ALE to DS/E Rise	t_{ASDS}	90	–	–	ns
Address Setup Time (R/$\overline{W}$)	t_{AS1}	15	–	–	ns
Address Setup Time ($\overline{CE}$)	t_{AS2}	55	–	–	ns
Address Hold Time (R/$\overline{W}$, $\overline{CE}$)	t_{AH}	10	–	–	ns
Muxed Address Valid Time to AS/ALE Fall	t_{ASL}	50	–	–	ns
Muxed Address Hold Time	t_{AHL}	20	–	–	ns
Peripheral Data Setup Time	t_{DSW}	195	–	–	ns
Write Data Hold Time	t_{DHW}	0	–	–	ns
Peripheral Output Data Delay Time From DS/E or $\overline{RD}$	t_{DDR}	–	–	220	ns
Read Data Hold Time	t_{DHR}	10	–	–	ns

CONTROL SIGNAL TIMING

Item		Symbol	min	typ	max	Unit
Oscillator Startup	1 MHz, 4 MHz	t_{RC}	–	–	100	ms
	32 kHz		–	–	1000	
Reset Pulse Width		t_{RWL}	5.0	–	–	μs
Reset Delay Time		t_{RLH}	5.0	–	–	μs
Power Sense Pulse Width		t_{PWL}	5.0	–	–	μs
Power Sense Delay Time		t_{PLH}	5.0	–	–	μs
$\overline{IRQ}$ Release from DS		t_{IRDS}	–	–	2.0	μs
$\overline{IRQ}$ Release from $\overline{RES}$		t_{IRR}	–	–	2.0	μs
VRT Bit Delay		t_{VRTD}	–	–	2.0	μs

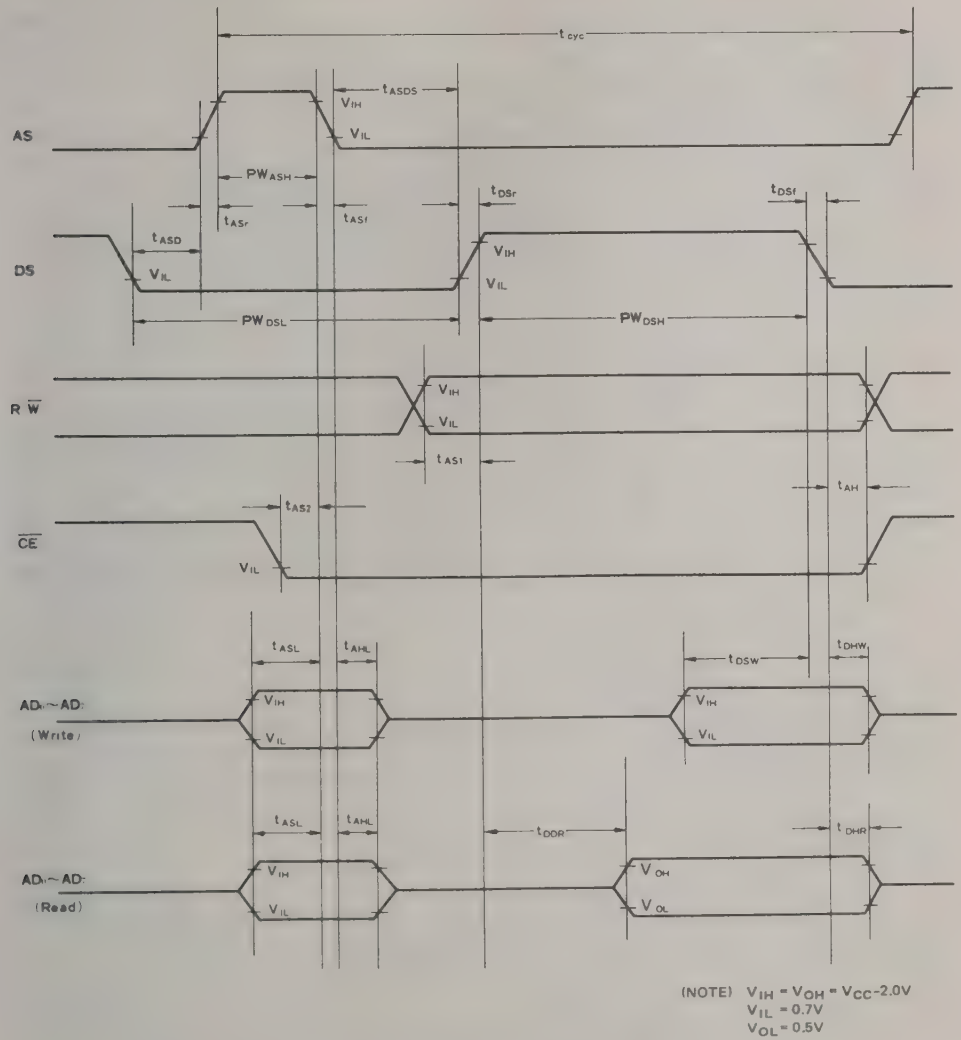


Figure 1 Bus Read, Write Timing (6801 Family)

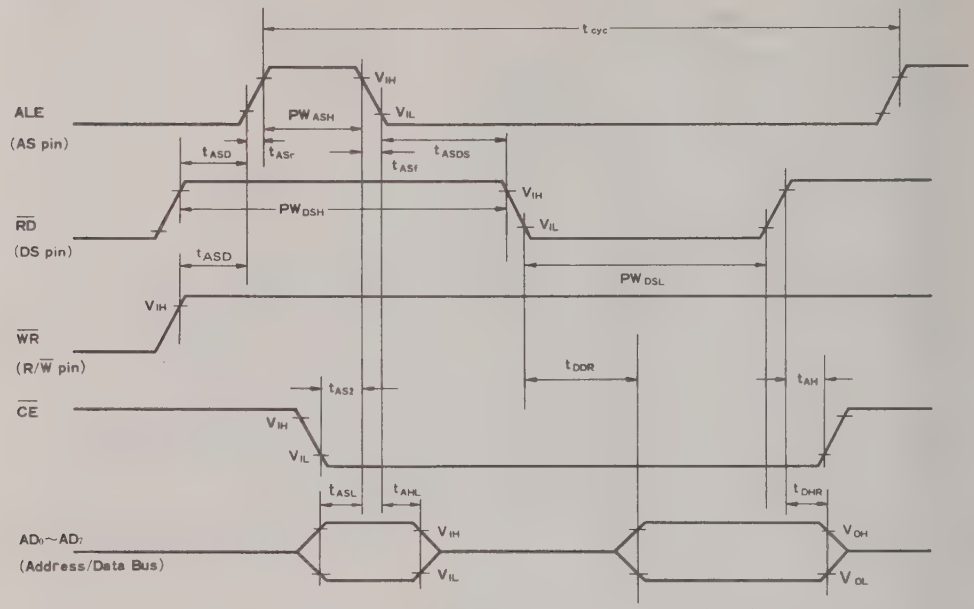
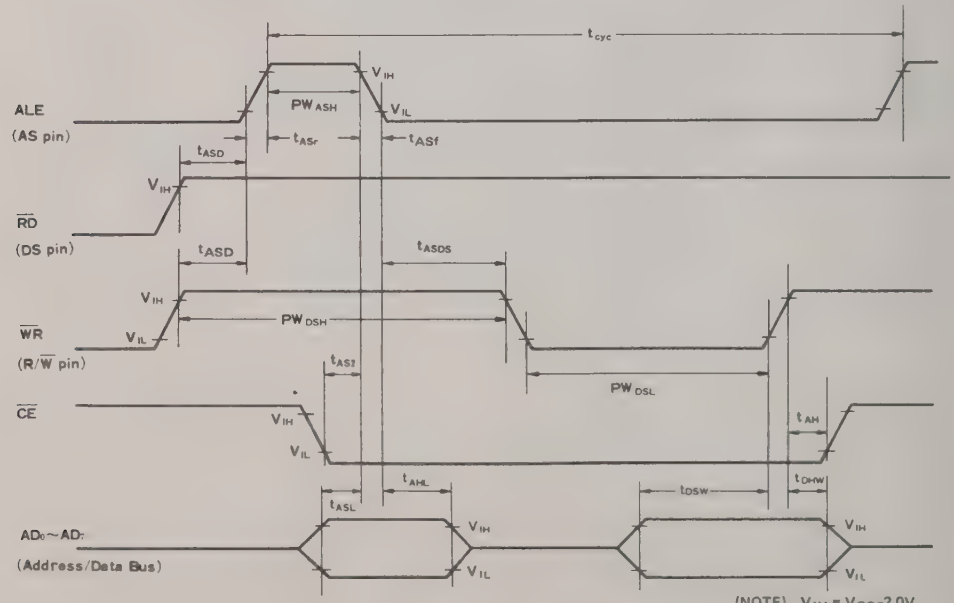


Figure 2 Read Timing (8085 Family)

(NOTE) $V_{IH} = V_{OH} = V_{CC}-2.0V$
$V_{IL} = 0.7V, V_{OL} = 0.5V$



(NOTE) $V_{IH} = V_{CC}-2.0V$
$V_{IL} = 0.7V$

Figure 3 Write Timing (8085 Family)

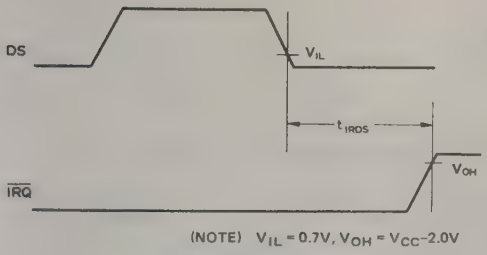


Figure 4 $\overline{IRQ}$ Release Delay (from DS)

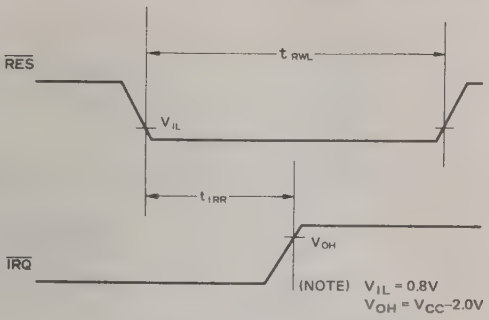
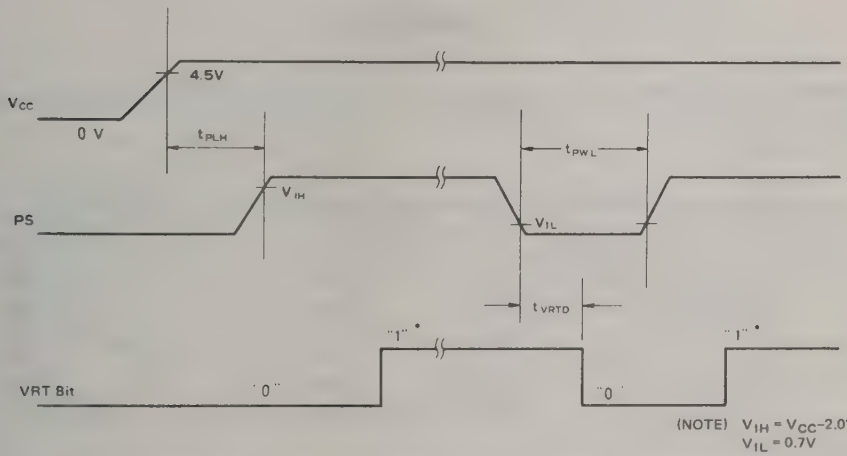


Figure 5 $\overline{IRQ}$ Release Delay (from $\overline{RES}$)



* The VRT bit is set to a "1" by reading control register #D. There is no additional way to clear the VRT bit.

Figure 6 VRT Bit Clear Timing

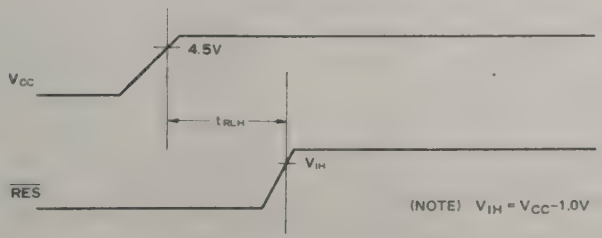


Figure 7 $\overline{RES}$ Release Delay

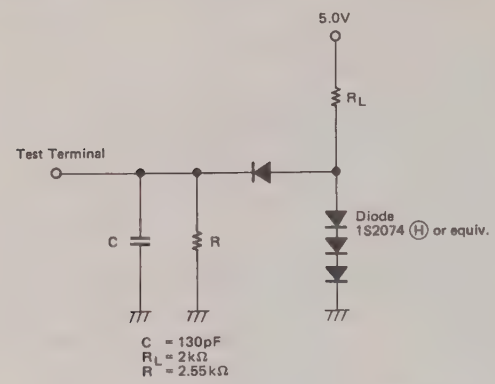


Figure 8 Test Load

■ BATTERY BACKED-UP OPERATION

• DEFINITION OF BATTERY BACKED-UP OPERATION

Active functions
(1) Clock function
(2) Retention of RAM data
(3) $\overline{RES}$, $\overline{IRQ}$, CKFS, CKOUT, PS, SQW functions

Inactive functions
(1) Data bus read/write operation

• BATTERY BACKED-UP ELECTRICAL CHARACTERISTICS (V_{SS} = 0V, T_a = 0 ~ +70°C, unless otherwise noted.)

Item		Symbol	Test Condition		min	typ	max	Unit
Supply Voltage		V_{CCL}			2.7	–	4.5	V
Supply Current	Crystal Oscillation	I_{CCL}*	V_{CCL} = 3.0V ~ SQW : disable CKOUT: fosc (No load)	4MHz	–	–	600	μA
				1MHz	–	–	350	μA
				32kHz	–	50	100	μA
	External Clock		V_{CCL} = 3.0V SQW : disable CKOUT: fosc (No load)	4MHz	–	–	500	μA
				1MHz	–	–	150	μA
				32kHz	–	30	70	μA
Battery Backed-up Transit Setup Time		t_{CE}	Fig. 9		0	–	–	ns
Operation Recovery Time		t_R			t_{cyc}	–	–	ns
Supply Voltage Fall Time		t_{Pf}			300	–	–	μs
Supply Voltage Rise Time		t_{Pr}			300	–	–	μs
Input "High" Voltage		V_{IHL}	V_{CCL} = 2.7V ~ 3.5V	$\overline{CE}$, PS CKFS	0.7×V_{CCL}	–	V_{CCL}	V
			V_{CCL} = 3.5V ~ 4.5V		2.5	–	V_{CCL}	V
				$\overline{RES}$	0.8×V_{CCL}	–	V_{CCL}	V
				OSC_1	0.8×V_{CCL}	–	V_{CCL}	V
Input "Low" Voltage		V_{ILL}		CKFS, PS	-0.3	–	0.5	V
				$\overline{RES}$	-0.3	–	0.5	V
				OSC_1	-0.3	–	0.5	V
Output "High" Voltage		V_{OHL}	I_{OH} = -800μA	SQW, CKOUT	0.8×V_{CCL}	–	–	V
Output "Low" Voltage		V_{OLL}	I_{OL} = 800μA	SQW, CKOUT	–	–	0.5	V
				$\overline{IRQ}$	–	–	0.5	V

* The time-base frequency to be used needs to be chosen in Register A.

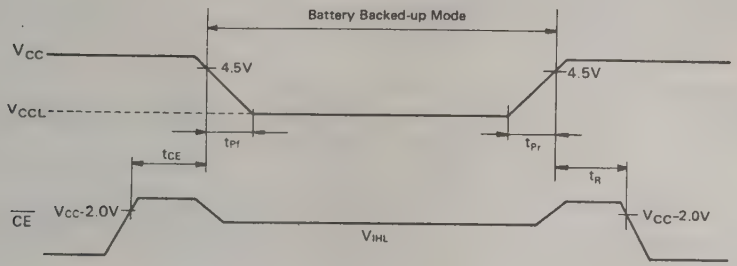


Figure 9 Battery Backed-up Timing

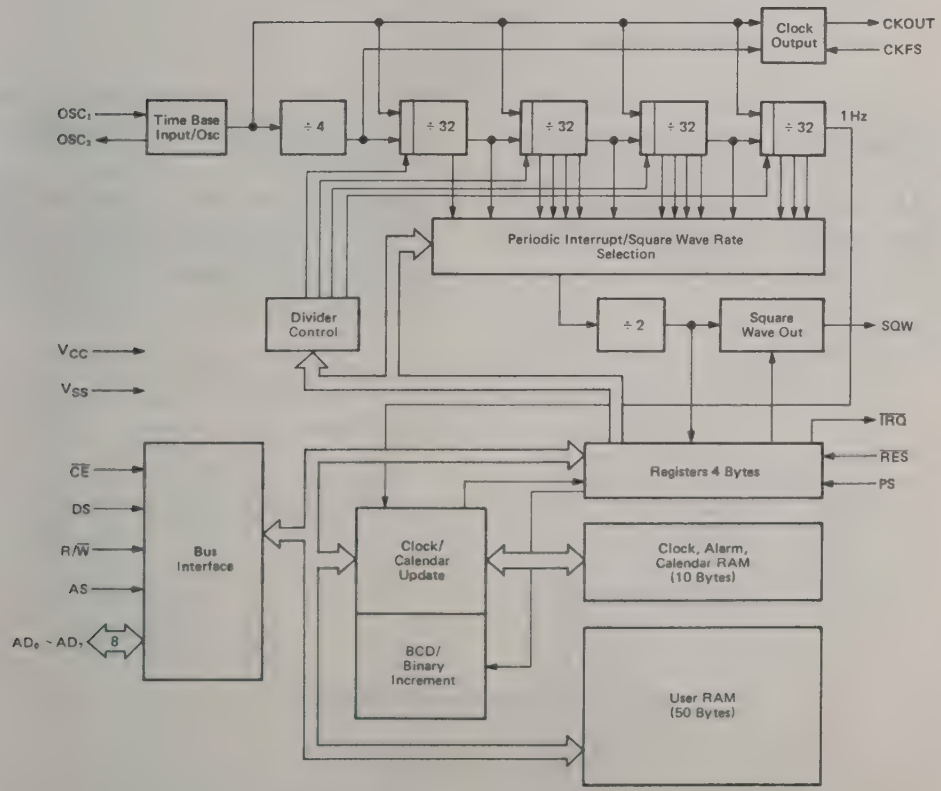


Figure 10 Block Diagram

■ CRYSTAL OSCILLATION CIRCUIT

The on-chip oscillator is designed for a parallel resonant crystal at 4.194304 MHz or 1.048576 MHz or 32.768 kHz frequencies. The crystal connections are shown in Figure 11.

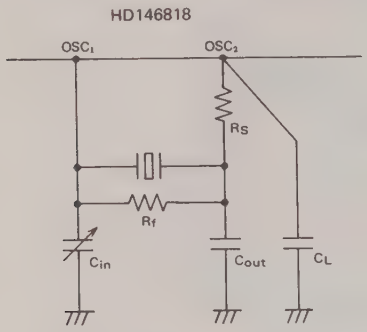


Figure 11 Crystal Oscillator Connection

Table 1 Oscillator Circuit Parameters

Parameter \ fosc	4.194304 MHz	1.048576 MHz	32.768 kHz
R_S	–	–	150 kΩ
R_f	150 kΩ	150 kΩ	5.6 MΩ
C_{in}	22 pF	33 pF	15 pF
C_{out}	22 pF	33 pF	33 pF
C_L	–	–	33 pF
CI	80 Ω (max)	700 Ω (max)	40 kΩ (max)

(NOTE) 1. R_S, C_L are used for 32.768 kHz only.
2. Capacitance (C_{in}) should be adjusted to accurate frequency. Parameters listed above are applied to the supply current measurement (See table of DC CHARACTERISTICS).
3. CI: Crystal Impedance

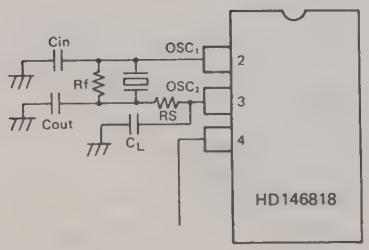
■ NOTE FOR BOARD DESIGN OF THE OSCILLATION CIRCUIT

In designing the board, the following notes should be taken when the crystal oscillator is used.

(1) Crystal oscillator, load capacity C_{in}, C_{out}, C_L and R_f, R_S must be placed near the LSI as much as possible.
[Normal oscillation may be disturbed when external noise is induced to pin 2 and 3.]

(2) Pin 3 signal line should be wired apart from pin 4 signal line as much as possible. Don't wire them in parallel, or normal oscillation may be disturbed when this signal is feedbacked to OSC_1.

(3) A signal line or a power source line must not cross or go near the oscillation circuit line as shown in the right figure to prevent the induction from these lines and perform the correct oscillation. The resistance among OSC_1, OSC_2 and other pins should be over 10MΩ.



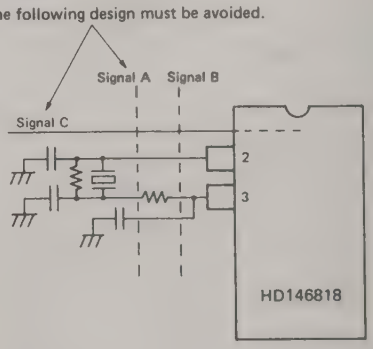


Figure 12 Note for Board Design of the Oscillation Circuit

■ INTERFACE CIRCUIT FOR HD6801, HD6301 AND 8085 PROCESSOR

HD146818 has a new interface circuit which permits the HD146818 to be directly interfaced with many type of multiplexed bus microprocessor such as HD6801, HD6301 and 8085 etc.

Figure 13 shows the bus control circuit. This circuit automatically selects the processor type by using AS/ASE to latch the state of DS/$\overline{RD}$ pin. Since DS is always "Low" and $\overline{RD}$ is always "High during AS/ALE, the latch automatically indicates which processor type is connected.

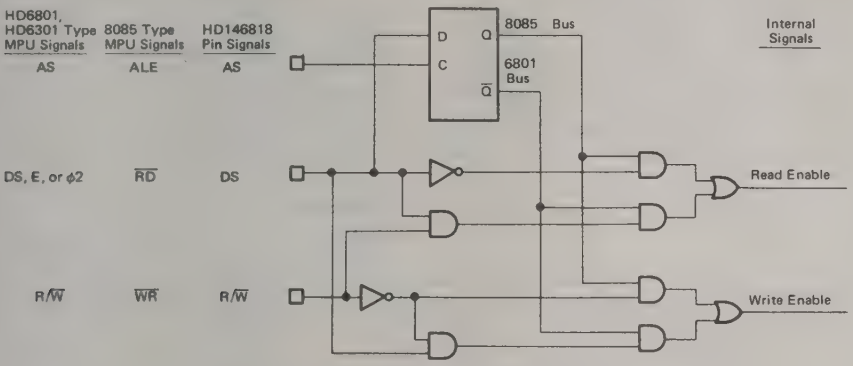


Figure 13 Functional Diagram of the Bus Control Circuit

■ ADDRESS MAP

Figure 14 shows the address map of the HD146818. The memory consists of 50 general purpose RAM bytes, 10 RAM bytes which normally contain the time, calendar, and alarm data, and four control and status bytes. All 64 bytes are directly readable and writable by the processor program except Registers C and D which are read only. Bit 7 of Register A and the seconds byte are also read only. Bit 7, of the second byte, always reads "0". The contents of the four control and status registers are described in the Register section.

• Time, Calendar, and Alarm Locations

The processor program obtains time and calendar information by reading the appropriate locations. The program may initialize the time, calendar, and alarm by writing to these RAM locations. The contents of the 10 time, calendar, and alarm byte may be either binary or binary-coded decimal (BCD).

Before initializing the internal registers, the SET bit in Register B should be set to a "1" to prevent time/calendar updates from occurring. The program initializes the 10 locations in the selected format (binary or BCD), then indicates the format in the data mode (DM) bit of Register B. All 10 time, calendar, and alarm bytes must use the same data mode, either binary or BCD. The SET bit may now be cleared to allow updates. Once initialized the real-time clock makes all updates in the selected data mode. The data mode cannot be changed without reinitializing the 10 data bytes.

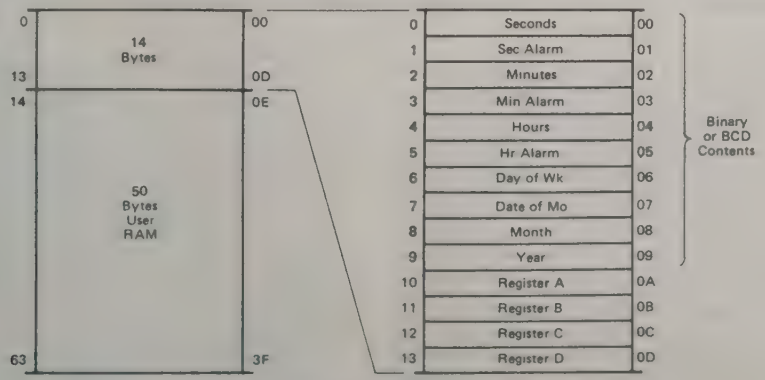


Figure 14 Address Map

Table 2 shows the binary and BCD formats of the 10 time, calendar, and alarm locations. The 24/12 bit in Register B establishes whether the hour locations represent 1-to-12 or 0-to-23. The 24/12 bit cannot be changed without reinitializing the hour locations. When the 12-hour format is selected the high-order bit of the hours byte represents PM when it is a "1".

The time, calendar, and alarm bytes are not always accessable by the processor program. Once-per-second the 10 bytes are switched to the update logic to be advanced by one second and to check for an alarm condition. If any of the 10 bytes are read at this time, the data outputs are undefined. The update lockout time is 248 μs at the 4.194304 MHz and 1.048567 MHz time bases and 1948 μs for the 32.768 kHz time base. The Update Cycle section shows how to accommodate the update cycle in the processor program.

The three alarm bytes may be used in two ways. When the program inserts an alarm time in the appropriate hours, minutes, and seconds alarm locations, the alarm interrupt is initiated at the specified time each day if the alarm enable bit is "1". The alternate usage is to insert a "don't care" state in one or more of three alarm bytes. The "don't care" code is any hexadecimal byte from C0 to FF. That is, the two most-significant bits of each byte, when set to "1", create a "don't care" situation. An alarm interrupt each hour is created with a "don't care" code in the hours alarm location. Similarly, an alarm is generated every minute with "don't care" codes in the hours and minutes alarm bytes. The "don't care" codes in all three alarm bytes create an interrupt every second.

Table 2 Time, Calendar, and Alarm Data Modes

Address Location	Function	Decimal Range	Range		Example*	
			Binary Data Mode	BCD Data Mode	Binary Data Mode	BCD Data Mode
0	Seconds	0~59	$00~$3B	$00~$59	15	21
1	Seconds Alarm	0~59	$00~$3B	$00~$59	15	21
2	Minutes	0~59	$00~$3B	$00~$59	3A	58
3	Minutes Alarm	0~59	$00~$3B	$00~$59	3A	58
4	Hours (12 Hour Mode)	1~12	$01~$0C (AM) and $81~$8C (PM)	$01~$12 (AM) and $81~$92 (PM)	05	05
	Hours (24 Hour Mode)	0~23	$00~$17	$00~$23	05	05
5	Hours Alarm (12 Hour Mode)	1~12	$01~$0C (AM) and $81~$8C (PM)	$01~$12 (AM) and $81~$92 (PM)	05	05
	Hours Alarm (24 Hour Mode)	0~23	$00~$17	$00~$23	05	05
6	Day of the Week Sunday = 1	1~7	$01~$07	$01~$07	05	05
7	Day of the Month	1~31	$01~$1F	$01~$31	0F	15
8	Month	1~12	$01~$0C	$01~$12	02	02
9	Year	0~99**	$00~$63	$00~$99	4F	79

* Example: 5:58:21 Thursday 15th February 1979

** Set the lower two digits of year in AD. If this number is multiple of 4, update applied to leap year is excuted.

• Static CMOS RAM

The 50 general purpose RAM bytes are not dedicated within the HD146818. They can be used by the processor program, and are fully available during the update cycle.

When time and calendar information must use battery back-up, very frequently there is other non-volatile data that must be retained when main power is removed. The 50 user RAM bytes serve the need for low-power CMOS battery-backed storage, and extend the RAM available to the program.

When further CMOS RAM is needed, additional HD146818s may be included in the system. The time/calendar functions may be disabled by holding the dividers, in Register A, in the reset state by setting the SET bit in Register B or by removing the oscillator. Holding the dividers in reset prevents interrupts or SQW output from operating while setting the SET bit allows these functions to occur. With the dividers clear, the available user RAM is extended to 59 bytes. Bit 7 of Register A, Registers C and D, and the high-order Bit of the seconds byte cannot effectively be used as general purpose RAM.

■ INTERRUPTS

The RTC plus RAM includes three separate fully automatic sources of interrupts to the processor. The alarm interrupt may be programmed to occur at rates from once-per-second to one-a-day. The periodic interrupt may be selected for rates from half-a-second to 30.517 μs. The update-ended interrupt may be used to indicate to the program that an up-date cycle is completed. Each of these independent interrupt conditions are described in greater detail in other sections.

The processor program selects which interrupts, if any, it wishes to receive. Three bits in Register B enable the three interrupts. Writing a "1" to a interrupt-enable bit permits that interrupt to be initiated when the event occurs. A "0" in the interrupt-enable bit prohibits the $\overline{IRQ}$ pin from being asserted due to the interrupt cause.

If an interrupt flag is already set when the interrupt becomes enabled, the $\overline{IRQ}$ pin is immediately activated, though the interrupt initiating the event may have occurred much earlier. Thus, there are cases where the program should clear such

earlier initiated interrupts before first enabling new interrupts.

When an interrupt event occurs a flag bit is set to a "1" in Register C. Each of the three interrupt sources have separate flag bits in Register C, which are set independent of the state of the corresponding enable bits in Register B. The flag bit may be used with or without enabling the corresponding enable bits.

In the software scanned case, the program does not enable the interrupt. The "interrupt" flag bit becomes a status bit, which the software interrogates, when it wishes. When the software detects that the flag is set, it is an indication to software that the "interrupt" event occurred since the bit was last read.

However, there is one precaution. The flag bits in Register C are cleared (record of the interrupt event is erased) when Register C is read. Double latching is included with Register C so the bits which are set are stable throughout the read cycle. All bits which are high when read by the program are cleared, and new interrupts (on any bits) are held until after the read cycle. One, two, or three flag bits may be found to be set when Register C is read. The program should inspect all utilized flag bits every time Register C is read to insure that no interrupts are lost.

The second flag bit usage method is with fully enabled interrupts. When an interrupt-flag bit is set and the corresponding interrupt-enable bit is also set, the $\overline{IRQ}$ pin is asserted "Low". $\overline{IRQ}$ is asserted as long as at least one of the three interrupt sources has its flag and enable bits both set. The IRQF bit in Register C is a "1" whenever the $\overline{IRQ}$ pin is being driven "Low".

The processor program can determine that the RTC initiated the interrupt by reading Register C. A "1" in bit 7 (IRQF bit) indicates that one of more interrupts have been initiated by the part. The act of reading Register C clears all the then-active flag bits, plus the IRQF bit. When the program finds IRQF set, it should look at each of the individual flag bits in the same byte which have the corresponding interrupt-mask bits set and service each interrupt which is set. Again, more than one interrupt-flag bit may be set.

■ DIVIDER STAGES

The HD146818 has 22 binary-divider stages following the time base as shown in Figure 10. The output of the dividers is a 1 Hz signal to the update-cycle logic. The dividers are controller by three divider bus (DV2, DV1, and DV0) in Register A.

● Divider Control

The divider-control bits have three uses, as shown in Table 3. Three usable operating time bases may be selected (4.194304 MHz, 1.048576 MHz, or 32.768 kHz). The divider chain may be held reset, which allows precision setting of the time. When the divider is changed from reset to an operating time base, the first update cycle is one second later. The divider-control bits are also used to facilitate testing the HD146818.

Table 3 Divider Configurations

Time-Base Frequency	Divider Bits Register A			Operation Mode	Divider Reset	Bypass First N-Divider Bits
	DV2	DV1	DV0			
4.194304 MHz	0	0	0	Yes	–	N = 0
1.048576 MHz	0	0	1	Yes	–	N = 2
32.768 kHz	0	1	0	Yes	–	N = 7
Any	1	1	0	No	Yes	–
Any	1	1	1	No	Yes	–

(NOTE) Other combinations of divider bits are used for test purposes only.

● Square-Wave Output Selection

Fifteen of the 22 divider taps are made available to a 1-of-15 selector as shown in Figure 10. The first purpose of selecting a divider tap is to generate a square-wave output signal in the SQW pin. Four bits in Register A establish the square-wave frequency as listed in Table 4. The SQW frequency selection shares the 1-of-15 selector with periodic interrupts.

Once the frequency is selected, the output of the SQW pin may be turned on and off under program control with the square-wave enable (SQWE) bit in Register B. Altering the divider, square-wave output selection bits, or the SQW output-enable bit may generate an asymetrical waveform at the time of execution. The square-wave output pin has a number of potential uses. For example, it can serve as a frequency standard for external use, a frequency synthesizer, or could be used to generate one or more audio tones under program control.

● Periodic Interrupt Selection

The periodic interrupt allows the $\overline{IRQ}$ pin to be triggered from once every 500 ms to once every 30.517 μs. The periodic interrupt is separate from the alarm interrupt which may be output from once-per-second to once-per-day.

Table 4 shows that the periodic interrupt rate is selected with the same Register A bits which select the square-wave frequency. Changing one also changes the other. But each function may be separately enabled so that a program could switch between the two features or use both. The SQW pin is enabled by the SQWE bit. Similarly the periodic interrupt is enabled by the PIE bit in Register B.

Periodic interrupt is usable by practically all real-time systems. It can be used to scan for all forms of input from contact closures to serial receive bits or tyes. It can be used in multiplexing displays or with software counters to measure inputs, create output intervals, or await the next needed software function.

Table 4 Periodic Interrupt Rate and Square Wave Output Frequency

Rate Select Control Register 1				4.194304 or 1.048576 MHz Time Base		32.768 kHz Time Base	
RS3	RS2	RS1	RS0	Periodic Interrupt Rate t_{PI}	SQW Output Frequency	Periodic Interrupt Rate t_{PI}	SQW Output Frequency
0	0	0	0	None	None	None	None
0	0	0	1	30.517 μs	32.768 kHz	3.90625 ms	256 Hz
0	0	1	0	61.035 μs	16.384 kHz	7.8125 ms	128 Hz
0	0	1	1	122.070 μs	8.192 kHz	122.070 μs	8.192 kHz
0	1	0	0	244.141 μs	4.096 kHz	244.141 μs	4.096 kHz
0	1	0	1	488.281 μs	2.048 kHz	488.281 μs	2.048 kHz
0	1	1	0	976.562 μs	1.024 kHz	976.562 μs	1.024 kHz
0	1	1	1	1.953125 ms	512 Hz	1.953125 ms	512 Hz
1	0	0	0	3.90625 ms	256 Hz	3.90625 ms	256 Hz
1	0	0	1	7.8125 ms	128 Hz	7.8125 ms	128 Hz
1	0	1	0	15.625 ms	64 Hz	15.625 ms	64 Hz
1	0	1	1	31.25 ms	32 Hz	31.25 ms	32 Hz
1	1	0	0	62.5 ms	16 Hz	62.5 ms	16 Hz
1	1	0	1	125 ms	8 Hz	125 ms	8 Hz
1	1	1	0	250 ms	4 Hz	250 ms	4 Hz
1	1	1	1	500 ms	2 Hz	500 ms	2 Hz

- **Initialization of the Time and the Start Sequence**

The first update of the time occurs about 500ms later after the SET bit of control register B is reset. So keep followings in mind when initializing and adjusting the time.

Procedure of time initialization

(1) Set the SET bit of control register B. (SET = "1")
(2) Set "1" into all the DV0, 1, 2 bits of control register A. (DV0 = DV1 = DV2 = "1")
(3) Set the time and calendar to each RAM.
(4) Set the frequency in use into DV0, 1 and DV2.
(5) Reset the SET bit. (SET = "0")

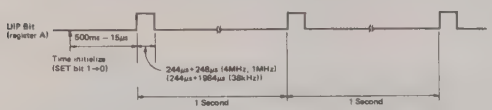


Figure 15 Time Initialization and the First Update

Restriction on Time-of-day and Calendar Initialization

There is a case in HD146818 (RTC) that update is not executed correctly if time of day and calendar shown below are initialized. Therefore, initialize the RTC without using time of day shown below.

Calendar, Time of day & Status after Update	Examples
If 29th 23:59:59 in all the months is initialized, update to 1st in the next month is executed. (Jan. – Dec. However except for Feb. 29th in leap year)	Mar. 29th →Apr. 1st
If 30th 23:59:59 in Apr., June, Sept., and Nov. is initialized, update to 31st in each month is executed.	Apr. 30th →Apr. 31st
If Feb. 28th 23:59:59 (not in leap year) is initialized, update to Feb. 29th is executed.	Feb. 28th,1983 →Feb. 29th,1983
If Feb. 28th 23:59:58 (in leap year) is initialized, update to Mar. 1st is executed.	Feb. 28th,1984 →Mar. 1st,1984

- **UPDATE CYCLE**

The HD146818 executes an update cycle once-per-second, assuming one of the proper time bases is in place, the divider is not clear, and the SET bit in Register B is clear. The SET bit in the "1" state permits the program to initialize the time and calendar bytes by stopping an existing update and preventing a new one from occurring.

The primary function of the update cycle is to increment the seconds byte, check for overflow, increment the minutes byte when appropriate and so forth through to the year of the century byte. The update cycle also compares each alarm byte with the corresponding time byte and issues an alarm if a match or if a "don't care" code (11XXXXXX) is present in all three positions.

With a 4.194304 MHz or 1.048576 MHz time base the up-

date cycle takes 248 μs while a 32.768 kHz time base update cycle takes 1984 μs. During the update cycle, the time, calendar, and alarm bytes are not accessable by the processor program. The HD146818 protects the program from reading transitional data. This protection is provided by switching the time, calendar, and alarm portion of the RAM off the microprocessor bus during the entire update cycle. If the processor reads these RAM locations before the update is complete the output will be undefined. The update in progress (UIP) status bit is set during the interval.

A program which randomly accesses the time and date information finds data unavailable statistically once every 4032 attempts. Three methods of accommodating nonavailability during update are usable by the program. In discussing the three methods it is assumed that at random points user programs are able to call a subroutine to obtain the time of day.

The first method of avoiding the update cycle uses the update-ended interrupt. If enabled, an interrupt occurs after every update cycle which indicates that over 999 ms are available to read valid time and date information. During this time a display could be updated or the information could be transfered to continuously available RAM. Before leaving the interrupt service routine, the IRQF bit in Register C should be cleared.

The second method uses the update-in-progress bit (UIP) in Register A to determine if the update cycle is in progress or not. The UIP bit will pulse once-per-second. Statistically, the UIP bit will indicate that time and date information is unavailable once every 2032 attempts. After the UIP bit goes "1", the update cycle begins 244 μs later. Therefore, if a "0" is read on the UIP bit, the user has at least 244 μs before the time/calendar data will be changed. If a "1" is read in the UIP bit, the time/calendar data may not be valid. The user should avoid interrupt service routines that would cause the time needed to read valid time/calendar data to exceed 244 μs.

The third method uses a periodic interrupt to determine if an update cycle is in progress. The UIP bit in Register A is set "1" between the setting of the PF bit in Register C (see Figure 16) Periodic interrupts that occur at a rate of greater than $t_{BUC} + t_{UC}$ allow valid time and date information to be read at each occurrence of the periodic interrupt. The reads should be completed within $(t_{PI} \div 2) + t_{BUC}$ to insure that data is not read during the update cycle.

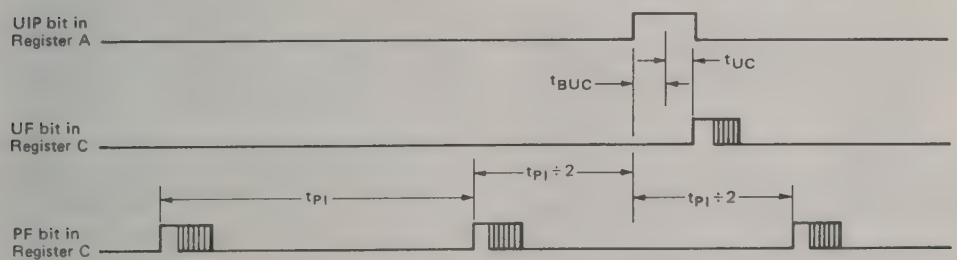
■ POWER-DOWN CONSIDERATIONS

In most systems, the HD146818 must continue to keep time when system power is removed. In such systems, a conversion from system power to an alternate power supply, usually a battery, must be made. During the transition from system to battery power, the designer of a battery backed-up RTC system must protect data integrity, minimize power consumption, and ensure hardware reliability according to the specification described in the section regarding Battery Backed-up operation.

The chip enable ($\overline{CE}$) pin controls all bus inputs (R/$\overline{W}$, DS, AS, $AD_0 \sim AD_7$). $\overline{CE}$, when negated, disallows any unintended modification of the RTC data by the bus. $\overline{CE}$ also reduces power consumption by reducing the number of transitions seen internally.

Power consumption may be further reduced by removing resistive and capacitive loads from the clock out (CKOUT) pin and the squarewave (SQW) pin.

During and after the power source conversion, the V_{IN} maximum specification must never be exceeded. Failure to meet the V_{IN} maximum specification can cause a virtual SCR to appear which may result in excessive current drain and destruction of the part.



t_{PI} = Periodic Interrupt Time Interval (500 ms, 250 ms, 125 ms, 62.5 ms, etc.)
t_{UC} = Update Cycle Time (248 μs or 1984 μs)
t_{BUC} = Delay Time Before Update Cycle (244 μs)

Figure 16 Update-Ended and Periodic Interrupt Relationship

■ SIGNAL DESCRIPTIONS

The block diagram in Figure 10, shows the pin connection with the major internal functions of the HD146818 Real-Time Clock plus RAM. The following paragraphs describe the function of each pin.

• V_{CC}, V_{SS}

DC power is provided to the part on these two pins, V_{CC} being the most positive voltage. The minimum and maximum voltages are listed in the Electrical Characteristics tables.

• OSC_1, OSC_2 – Time Base (Inputs)

The time base for the time functions may be an external signal or the crystal oscillator. External square waves at 4.194304 MHz, 1.048576 MHz, or 32.768 kHz may be connected to OSC_1 as shown in Figure 17. The time-base frequency to be used is chosen in Register A.

The on-chip oscillator is designed for a parallel resonant crystal at 4.194304 MHz or 1.048576 MHz or 32.768 kHz frequencies. The crystal connections are shown in Figure 11.

• CKOUT – Clock Out (Output)

The CKOUT pin is an output at the time-base frequency divided by 1 or 4. A major use for CKOUT is as the input clock to the microprocessor; thereby saving the cost of a second crystal. The frequency of CKOUT depends upon the time-base frequency and the state of the CKFS pin as shown in Table 5.

• CKFS – Clock Out Frequency Select (Input)

The CKOUT pin is an output at the time-base frequency divided by 1 or 4. CKFS tied to V_{CC} causes CKOUT to be the same frequency as the time base at the OSC_1 pin. When CKFS is at V_{SS}, CKOUT is the OSC_1 time-base frequency divided by four. Table 5 summarizes the effect of CKFS.

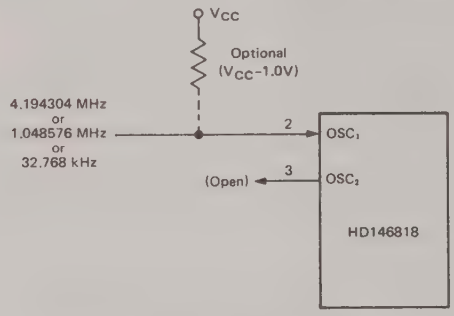


Figure 17 External Time-Base Connection

Table 5 Clock Output Frequencies

Time Base (OSC_1) Frequency	Clock Frequency Select Pin (CKFS)	Clock Frequency Output Pin (CKOUT)
4.194304 MHz	"High"	4.194304 MHz
4.194304 MHz	"Low"	1.048576 MHz
1.048576 MHz	"High"	1.048576 MHz
1.048576 MHz	"Low"	262.144 kHz
32.768 kHz	"High"	32.768 kHz
32.768 kHz	"Low"	8.192 kHz

• SQW – Square Wave (Output)

The SQW pin can output a signal one of 15 of the 22 internal-divider stages. The frequency and output enable of the SQW may be altered by programming Register A, as shown in Table 4. The SQW signal may be turned on and off using a bit in Register B.

• $AD_0 \sim AD_7$ – Multiplexed Bidirectional Address/Data Bus

Multiplexed bus processors save pins by presenting the address during the first portion of the bus cycle and using the same pins during the second portion for data. Address-then-data multiplexing does not slow the access time of the HD146818 since the bus reversal from address to data is occurring during the internal RAM access time.

The address must be valid just prior to the fall of AS/ALE at which time the HD146818 latches the address from AD_0 to AD_5. Valid write data must be presented and held stable during the latter portion of the DS or $\overline{WR}$ pulses. In a read cycle, the HD146818 outputs 8 bits of data during the latter portion of the DS or $\overline{RD}$ pulses, then ceases driving the bus (returns the output drivers to three-state) when DS falls in the HD6801, HD6301 case or $\overline{RD}$ rises in the other case.

• AS – Multiplexed Address Strobe (Input)

A positive going multiplexed address strobe pulse serves to demultiplex the bus. The falling edge of AS or ALE causes the address to be latched within the HD146818. The bus control circuit in the HD146818 also latches the state of the DS pin with the falling edge of AS or ALE.

• DS – Data Strobe or Read (Input)

The DS pin has two interpretations via the bus control circuit. When emanating from 6801 family type processor, DS is a positive pulse during the latter portion of the bus cycle, and is variously called DS (data strobe), E (enable), and ϕ_2 (ϕ_2 clock). During read cycles, DS signifies the time that the RTC is to drive the bidirectional bus. In write cycles, the trailing edge of DS causes the Real-Time Clock plus RAM to latch the written data.

The second interpretation of DS is that of $\overline{RD}$, $\overline{MEMR}$, or $\overline{I/OR}$ emanating from the 8085 type processor. In this case, DS identifies the time period when the real-time clock plus RAM drives the bus with read data. This interpretation of

DS is also the same as an output-enable signal on a typical memory.

The bus control circuit, within the HD146818, latches the state of the DS pin on the falling edge of AS/ALE. When 6801 mode, DS must be "Low" during AS/ALE, which is the case with 6801 family multiplexed bus processors. To insure the 8085 mode of this circuit the DS pin must remain "High" during the time AS/ALE is "High".

- **R/$\overline{W}$ – Read/Write (Input)**

The bus control circuit treats the R/$\overline{W}$ pin in one of two ways. When 6801 family type processor is connected, R/$\overline{W}$ is a level which indicates whether the current cycle is a read or write. A read cycle is indicated with a "High" level on R/$\overline{W}$ while DS is "High", whereas a write cycle is a "Low" on R/$\overline{W}$ during DS

The second interpretation of R/$\overline{W}$ is as a negative write pulse, $\overline{WR}$, $\overline{MEMW}$, and $\overline{I/OW}$ from 8085 type processors. This circuit in this mode gives R/$\overline{W}$ pin the same meaning as the write ($\overline{W}$) pulse on many generic RAMs.

- **$\overline{CE}$ – Chip Enable (Input)**

The chip-enable ($\overline{CE}$) signal must be asserted ("Low") for a bus cycle in which the HD146818 is to be accessed. $\overline{CE}$ is not latched and must be stable during DS and AS (in the 6801 case) and during $\overline{RD}$ and $\overline{WR}$ (in the 8085 case). Bus cycles which take place without asserting $\overline{CE}$ cause no actions to take place within the HD146818. When $\overline{CE}$ is "High", the multiplexed bus output is in a high-impedance state.

When $\overline{CE}$ is "High", all address, data, DS, and R/$\overline{W}$ inputs from the processor are disconnected within the HD146818. This permits the HD146818 to be isolated from a powered-down processor. When $\overline{CE}$ is held "High", an unpowered device cannot receive power through the input pins from the real-time clock power source. Battery power consumption can thus be reduced by using a pullup resistor or active clamp on $\overline{CE}$ when the main power is off.

- **$\overline{IRQ}$ – Interrupt Request (Output)**

The $\overline{IRQ}$ pin is an active "Low" output of the HD146818 that may be used as an interrupt input to a processor. The $\overline{IRQ}$ output remains "Low" as long as the status bit causing the interrupt is present and the corresponding interrupt-enable bit is set. To clear the $\overline{IRQ}$ pin, the processor program normally reads Register C. The $\overline{RES}$ pin also clears pending interrupts.

When no interrupt conditions are present, the $\overline{IRQ}$ level is in the high-impedance state. Multiple interrupting devices may thus be connected to an $\overline{IRQ}$ bus with one pullup at the processor.

- **$\overline{RES}$ – Reset (Input)**

The $\overline{RES}$ pin does not affect the clock, calendar, or RAM functions. On powerup, the $\overline{RES}$ pin must be held "Low" for the specified time, t_{RLH}, in order to allow the power supply to stabilize, Figure 18 shows a typical representation of the $\overline{RES}$ pin circuit.

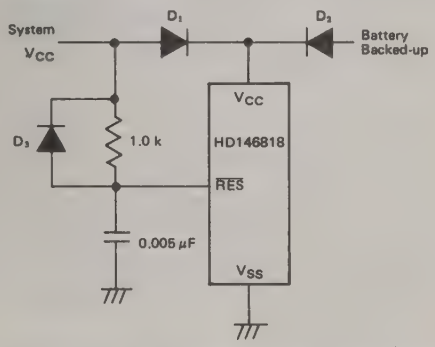
When $\overline{RES}$ is "Low" the following occurs:

a) Periodic Interrupt Enable (PIE) bit is cleared to "0".
b) Alarm Interrupt Enable (AIE) bit is cleared to "0".
c) Update ended interrupt Enable (UIE) bit is cleared to "0".
d) Update ended Interrupt Flag (UF) bit is cleared to "0".
e) Interrupt Request status Flag (IRQF) bit is cleared to "0".
f) Periodic Interrupt Flag (PF) bit is cleared to "0".
g) Alarm Interrupt Flag (AF) bit is cleared to "0".
h) $\overline{IRQ}$ pin is in high-impedance state, and
i) Square Wave output Enable (SQWE) bit is cleared to "0".

- **PS – Power Sense (Input)**

The power-sense pin is used in the control of the valid RAM and time (VRT) bit in Register C. When the PS pin is "Low" the VRT bit is cleared to "0".

During powerup, the PS pin must be externally held "Low" for the specified time, t_{PLH}. As power is applied the VRT bit remain "Low" indicating that the contents of the RAM, time registers, and calendar are not guaranteed. When normal operation commences PS should be permitted to go "High". Output signal from external power sence circuit will be connected to this input.



(NOTE) If the RTC is isolated from the MPU or MCU power by a diode drop, care must be taken to meet V_{IH} requirements.

Figure 18 Typical Powerup Delay Circuit for $\overline{RES}$

■ REGISTERS

The HD146818 has four registers which are accessible to the processor program. The four registers are also fully accessible during the update cycle.

- **Register A ($0A)**

MSB							LSB	
b7	b6	b5	b4	b3	b2	b1	b0	Read/Write Register except UIP
UIP	DV2	DV1	DV0	RS3	RS2	RS1	RS0	

UIP – The update in progress (UIP) bit is a status flag that may be monitored by the program. When UIP is a "1" the update

Table 6 Update Cycle Times

UIP Bit	Time Base (OSC_1)	Update Cycle Time (t_{UC})	Minimum Time Before Update Cycle (t_{BUC})
1	4.194304 MHz	248 μs	–
1	1.048576 MHz	248 μs	–
1	32.768 kHz	1984 μs	–
0	4.194304 MHz	–	244 μs
0	1.048576 MHz	–	244 μs
0	32.768 kHz	–	244 μs

cycle is in progress or will soon begin. When UIP is a "0" the update cycle is not in progress and will not be for at least 244 μs (for all time bases). This is detailed in Table 6. The time, calendar, and alarm information in RAM is fully available to the program when the UIP bit is zero – it is not in transition. The UIP bit is a read-only bit, and is not affected by Reset. Writing the SET bit in Register B to a "1" inhibit any update cycle and then clear the UIP status bit.

DV2, DV1, DV0 – Three bits are used to permit the program to select various conditions of the 22-stage divider chain. The divider selection bits identify which of the three time-base frequencies is in use. Table 3 shows that time bases of 4.194304 MHz, 1.048576 MHz, and 32.768 kHz may be used. The divider selection bits are also used to reset the divider chain. When the time/calendar is first initialized, the program may start the divider at the precise time stored in the RAM. When the divider reset is removed the first update cycle begins half a second later. These three read/write bits are never modified by the RTC and are not affected by $\overline{RES}$.

RS3, RS2, RS1, RS0 – The four rate selection bits select one of 15 taps on the 22-stage divider, or disable the divider output. The tap selected may be used to generate an output square wave (SQW pin) and/or a periodic interrupt. The program may do one of the following: 1) enable the interrupt with the PIE bit, 2) enable the SQW output pin with the SQWE bit, 3) enable both at the same time at the same rate, or 4) enable neither. Table 4 lists the periodic interrupt rates and the square-wave frequencies that may be chosen with the RS bits. These four bits are read/write bits which are not affected by $\overline{RES}$ and are never changed by the RTC.

- **Register B ($0B)**

MSB LSB

b7	b6	b5	b4	b3	b2	b1	b0
SET	PIE	AIE	UIE	SQWE	DM	24/12	DSE

Read/Write Register

SET – When the SET bit is a "0", the update cycle functions normally by advancing the counts once-per-second. When the SET bit is written to a "1", any update cycle in progress is aborted and the program may initialize the time and calendar bytes without an update occurring in the midst of initializing. SET is a read/write bit which is not modified by $\overline{RES}$ or internal functions of the HD146818.

PIE – The periodic interrupt enable (PIE) bit is a read/write bit which allows the periodic-interrupt flag (PF) bit to cause the $\overline{IRQ}$ pin to be driven "Low". A program writes a "1" to the PIE bit in order to receive periodic interrupts at the rate specified by the RS3, RS2, RS1, and RS0 bits in Control Register A. A "0" in PIE blocks $\overline{IRQ}$ from being initiated by a periodic interrupt, but the periodic flag (PF) bit is still at the periodic rate. PIE is not modified by any internal HD146818 functions, but is cleared to "0" by a $\overline{RES}$.

AIE – The alarm interrupt enable (AIE) bit is a read/write bit which when set to a "1" permits the alarm flag (AF) to assert $\overline{IRQ}$. An alarm interrupt occurs for each second that the three time bytes equal the three alarm bytes (including a "don't care" alarm code of binary 11XXXXXX). When the AIE bit is a "0", the AF bit does not initiate an $\overline{IRQ}$ signal. The $\overline{RES}$ pin clears AIE to "0". The internal functions do not affect the AIE bit.

UIE – The UIE (update-ended interrupt enable) bit is a read/write bit which enables the update-end flage (UF) bit to assert $\overline{IRQ}$. The $\overline{RES}$ pin going "Low" or the SET bit going "1" clears the UIE bit.

SQWE – When the square-wave enable (SQWE) bit is set to a "1" by the program, a square-wave signal at the frequency specified in the rate selection bits (RS3 to RS0) appears on the SQW pin. When the SQWE bit is set to a "0" the SQW pin is held "Low". The state of SQWE is cleared by the $\overline{RES}$ pin. SQWE is a read/write bit.

DM – The data mode (DM) bit indicates whether time and calendar updates are to use binary or BCD formats. The DM bit is written by the processor program and may be read by the program, but is not modified by any internal functions or $\overline{RES}$. A "1" in DM signifies binary data, while a "0" in DM specified binary-coded-decimal (BCD) data.

24/12 – The 24/12 control bit establishes the format of the hours bytes as either the 24-hour mode (a "1") or the 12-hour mode (a "0"). This is a read/write bit, which is affected only by the software.

DSE – The daylight savings enable (DSE) bit is a read/write bit which allows the program to enable two special updates (when DSE is a "1"). On the last Sunday in April the time increments from 1:59:59 AM to 3:00:00 AM. On the last Sunday in October when the time first reaches 1:59:59 AM it changes to 1:00:00 AM. These special updates do not occur when the DSE bit is a "0". DSE is not changed by any internal operations or reset.

- **Register C ($0C)**

MSB LSB

b7	b6	b5	b4	b3	b2	b1	b0
IRQF	PF	AF	UF	0	0	0	0

Read-Only Register

IRQF – The interrupt request flag (IRQF) is set to a "1" when one or more of the following are true:

PF = PIE = "1"
AF = AIE = "1"
UF = UIE = "1"

i.e., IRQF = PF · PIE + AF · AIE + UF · UIE

Any time the IRQF bit is a "1", the $\overline{IRQ}$ pin is driven "Low". All flag bits are cleared after Register C is read by the program or when the $\overline{RES}$ pin is low. A program write to Register C does not modify any of the flag bits.

PF – The periodic interrupt flag (PF) is a read-only bit which is set to a "1" when a particular edge is detected on the selected tap of the divider chain. The RS3 to RS0 bits establish the periodic rate. PF is set to a "1" independent of the state of the PIE bit. PF being a "1" initiates an $\overline{IRQ}$ signal and the IRQF bit when PIE is also a "1". The PF bit is cleared by a $\overline{RES}$ or a software read of Register C.

AF – A "1" in the AF (alarm interrupt flag) bit indicates that the current time has matched the alarm time. A "1" in the AF causes the $\overline{IRQ}$ pin to go "Low", and a "1" to appear in the IRQF bit, when the AIE bit also is a "1". A $\overline{RES}$ or a read

of Register C clears AF.

UF – The update-ended interrupt flag (UF) bit is set after each update cycle. When the UIE bit is a "1", the "1" in UF causes the IRQF bit to be a "1", asserting $\overline{IRQ}$. UF is cleared by a Register C read or a $\overline{RES}$.

b3 to b0 – The unused bits of Status Register C are read as "0's". They can not be written.

- **Register D ($0D)**

MSB LSB

b7	b6	b5	b4	b3	b2	b1	b0	Read Only Register
VRT	0	0	0	0	0	0	0	

VRT – The valid RAM and time (VRT) bit indicates the condition of the contents of the RAM, provided the power sense (PS) pin is satisfactorily connected. A "0" appears in the VRT bit when the power-sense pin is "Low". The processor program can set the VRT bit when the time and calendar are initialized to indicate that the RAM and time are valid. The VRT is a read/only bit which is not modified by the $\overline{RES}$ pin. The VRT bit can only be set by reading the Register D. For setting this bit, PS signal needs to be "High" level.

b6 to b0 – The remaining bits of Register D are unused. They cannot be written, but are always read as "0's".

■ NOTE FOR USE

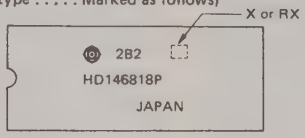
Input Signal, which is not necessary for user's application should be used fixed to "High" or "Low" level. This is applicable to the following signal pins.

CKFS, PS

RESTRICTION ON HD146818 USAGE (1)

The daylight saving function can not be performed on the HD146818P (X type). So do not use this function for the system design.

< Type number > HD146818P (X type . . . . . Marked as follows)



< Restriction on usage >

Please set "0" to DSE bit (Daylight Saving Enable bit) on initializing the control register B.
DSE = "1" is prohibited.

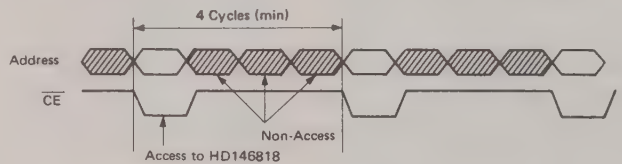
RESTRICTION ON HD146818 USAGE (2)

Access to HD146818 needs to be performed under following conditions.

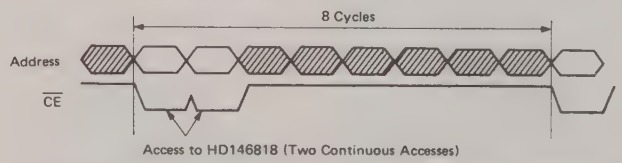
(i) Chip-enable ($\overline{CE}$) must be asserted to active "Low" level only when MPU performs read/write operation from/into internal RAM (Time and Calendar RAM, Control register, User RAM).

(ii) User RAM and control register must be accessed in less than 1/4 frequency shown below.
(Example: After one access, non-access cycles more than three cycles are necessary to be inserted.)

[Example 1]



[Example 2]



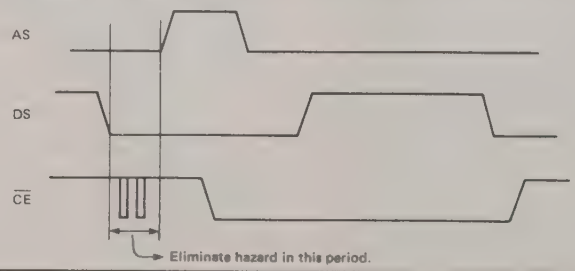
As shown in the above [example 2], when HD146818 is accessed continuously, continuous access must not be executed over fifty times.

(iii) The application that User RAM is used for program area should be avoided. (Inhibit continuous access.)

(iv) Minimize the noise by inserting noise bypass condenser between power supply and ground pin (V_{CC}-V_{SS}). (Insert noise bypass condenser as near HD146818 as possible.)

RESTRICTION ON HD146818 USAGE (3)

Chip-enable ($\overline{CE}$) must be stable between falling edge of DS and vising edge of AS shown below. (Address decoder hazard needs to be externally suppressed in this period.)



16-BIT MULTI-CHIP MICROCOMPUTER

HD68000, HD68000Y, HD68000P, HD68000PS
MPU (Micro Processing Unit)

The HD68000 is the first in a family of advanced microprocessors from Hitachi. Utilizing VLSI technology, the HD68000 is a fully-implemented 16-bit microprocessor with 32-bit registers, a rich basic instruction set, and versatile addressing modes.

The HD68000 possesses an asynchronous bus structure with a 24-bit address bus and a 16-bit data bus.

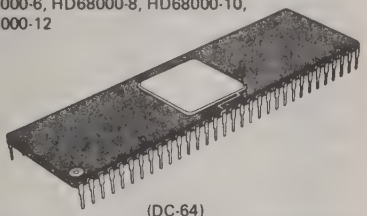
■ FEATURES

- 32-Bit Data and Address Registers
- 16 Megabyte Direct Addressing Range
- 56 Powerful Instruction Types
- Operations of Five Main Data Types
- Memory Mapped I/O
- 14 Addressing Modes

■ TYPE OF PRODUCTS

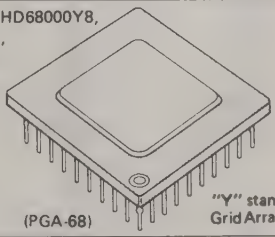
Type No.	Package	Clock Frequency (MHz)
HD68000-6	DC-64	6.0
HD68000-8		8.0
HD68000-10		10.0
HD68000-12		12.5
HD68000Y6	PGA-68	6.0
HD68000Y8		8.0
HD68000Y10		10.0
HD68000Y12		12.5
HD68000P6	DP-64	6.0
HD68000P8		8.0
HD68000PS6	DP-64S	6.0
HD68000PS8		8.0

HD68000-6, HD68000-8, HD68000-10, HD68000-12



(DC-64)

HD68000Y6, HD68000Y8, HD68000Y10, HD68000Y12



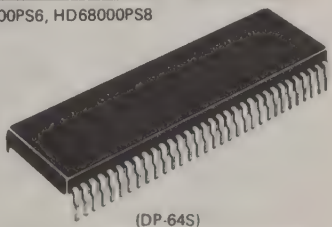
(PGA-68)

"Y" stands for Pin Grid Array Package.

HD68000P6, HD68000P8

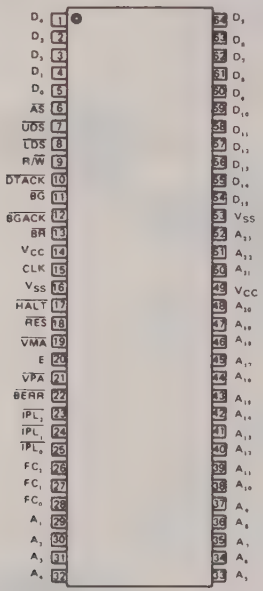
(DP-64)

HD68000PS6, HD68000PS8



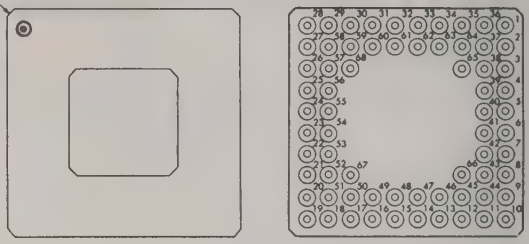
(DP-64S)

- **PIN ARRANGEMENT**
- HD68000, HD68000P, HD68000PS



(Top View)

- HD68000Y



(Bottom View)

Pin No.	Function	Pin No.	Function	Pin No.	Function	Pin No.	Function
1	N/C	18	A_9	35	D_1	52	A_{12}
2	$\overline{DTACK}$	19	N/C	36	$\overline{AS}$	53	A_{13}
3	$\overline{BGACK}$	20	A_{14}	37	$\overline{LDS}$	54	A_{18}
4	$\overline{BR}$	21	A_{15}	38	$\overline{BG}$	55	V_{CC}
5	CLK	22	A_{17}	39	V_{CC}	56	V_{SS}
6	$\overline{HALT}$	23	A_{19}	40	V_{SS}	57	A_{23}
7	$\overline{VMA}$	24	A_{20}	41	$\overline{RES}$	58	D_{14}
8	E	25	A_{21}	42	$\overline{VPA}$	59	D_{11}
9	$\overline{BERR}$	26	A_{22}	43	$\overline{IPL_2}$	60	D_9
10	N/C	27	D_{15}	44	$\overline{IPL_0}$	61	D_6
11	FC_2	28	D_{12}	45	FC_1	62	D_3
12	FC_0	29	D_{10}	46	N/C	63	D_0
13	A_1	30	D_8	47	A_2	64	$\overline{UDS}$
14	A_3	31	D_7	48	A_5	65	$R/\overline{W}$
15	A_4	32	D_5	49	A_6	66	$\overline{IPL_1}$
16	A_6	33	D_4	50	A_{10}	67	A_{13}
17	A_7	34	D_2	51	A_{11}	68	D_{13}

■ ABSOLUTE MAXIMUM RATINGS

Item	Symbol	Value	Unit
Supply Voltage	V_{CC}*	-0.3 ~ +7.0	V
Input Voltage	V_{in}*	-0.3 ~ +7.0	V
Operating Temperature Range	T_{opr}	0 ~ +70	°C
Storage Temperature	T_{stg}	-55 ~ +150	°C

* With respect to V_{SS} (SYSTEM GND)

(NOTE) Permanent LSI damage may occur if maximum ratings are exceeded. Normal operation should be under recommended operating conditions. If these conditions are exceeded, it could affect reliability of LSI.

■ RECOMMENDED OPERATING CONDITIONS

Item	Symbol	min	typ	max	Unit
Supply Voltage	V_{CC}*	4.75	5.0	5.25	V
Input Voltage	V_{IH}*	2.0	–	V_{CC}	V
	V_{IL}*	-0.3	–	0.8	V
Operating Temperature	T_{opr}	0	25	70	°C

* With respect to V_{SS} (SYSTEM GND)

■ ELECTRICAL CHARACTERISTICS

● DC CHARACTERISTICS (V_{CC} = 5V ±5%, V_{SS} = 0V, Ta = 0 ~ +70°C, Fig. 1, unless otherwise noted.)

Item		Symbol	Test Condition	min	max	Unit
Input "High" Voltage		V_{IH}		2.0	V_{CC}	V
Input "Low" Voltage		V_{IL}		V_{SS}-0.3	0.8	V
Input Leakage Current	$\overline{BERR}$, $\overline{BGACK}$, $\overline{BR}$, $\overline{DTACK}$, $\overline{IPL_0}$ ~ $\overline{IPL_2}$, VPA, CLK	I_{in}	@5.25V	–	2.5	μA
	$\overline{HALT}$, $\overline{RES}$			–	20	
Three-State (Off State) Input Current	$\overline{AS}$, A_1 ~ A_{23}, D_0 ~ D_{15}, FC_0 ~ FC_2, $\overline{LDS}$, R/$\overline{W}$, $\overline{UDS}$, $\overline{VMA}$	I_{TSI}	@2.4V/0.4V	–	20	μA
Output "High" Voltage	$\overline{AS}$, A_1 ~ A_{23}, $\overline{BG}$, D_0 ~ D_{15}, FC_0 ~ FC_2, $\overline{LDS}$, R/$\overline{W}$, $\overline{UDS}$, $\overline{VMA}$, E	V_{OH}	I_{OH} = -400μA	2.4	–	V
	E*			V_{CC}-0.75	–	
Output "Low" Voltage	$\overline{HALT}$	V_{OL}	I_{OL} = 1.6 mA	–	0.5	V
	A_1 ~ A_{23}, $\overline{BG}$, FC_0 ~ FC_2		I_{OL} = 3.2 mA	–	0.5	
	$\overline{RES}$		I_{OL} = 5.0 mA	–	0.5	
	$\overline{AS}$, D_0 ~ D_{15}, $\overline{LDS}$, R/$\overline{W}$, E, $\overline{UDS}$, $\overline{VMA}$		I_{OL} = 5.3 mA	–	0.5	
Power Dissipation	HD68000 HD68000Y	P_D	f=6MHz f=8MHz f=10MHz	–	1.5	W
			f=12.5MHz	–	1.75	
	HD68000P HD68000PS		f=8MHz, V_{CC}=5V, Ta=25°C	–	0.9	
Capacitance (Package Type Dependent)		C_{in}	V_{in} = 0V, Ta = 25°C, f = 1 MHz	–	20.0	pF

* With external pull up resistor of 1.1 kΩ.

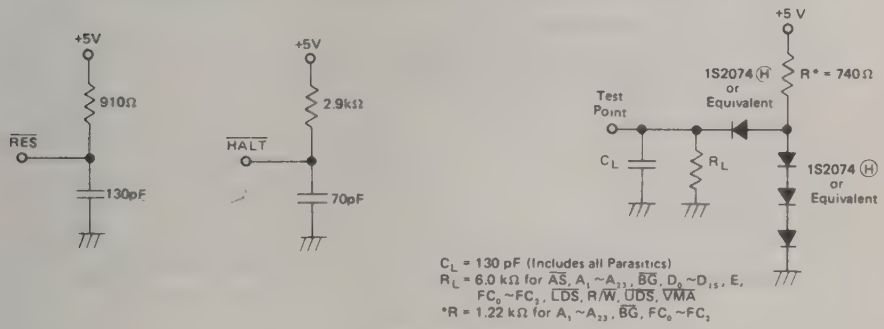
- **AC CHARACTERISTICS (V_{CC} = 5.0V ±5%, V_{SS} = 0V, Ta = 0 ~ +70°C, unless otherwise noted.)**

Number	Item	Symbol	Test Condition	6MHz Version HD68000-6 HD68000Y6 HD68000P6		8MHz Version HD68000-8 HD68000Y8 HD68000P8 HD68000PS8		10MHz Version HD68000-10 HD68000Y10		12.5MHz Version HD68000-12 HD68000Y12		Unit
				min	max	min	max	min	max	min	max	
	Frequency of Operation	f		4.0	6.0	4.0	8.0	4.0	10.0	4.0	12.5	MHz
①	Clock Period	t_{cyc}		167	250	125	250	100	250	80	250	ns
②	Clock Width "Low"	t_{CL}		75	125	55	125	45	125	35	125	ns
③	Clock Width "High"	t_{CH}		75	125	55	125	45	125	35	125	ns
④	Clock Fall Time	t_{Cf}		–	10	–	10	–	10	–	5	ns
⑤	Clock Rise Time	t_{Cr}		–	10	–	10	–	10		5	ns
⑥	Clock "Low" to Address	t_{CLAV}		–	80	–	70	–	60	–	55	ns
⑥A	Clock "High" to FC Valid	t_{CHFCV}		–	80	–	80	–	60	–	55	ns
⑦	Clock "High" to Address/Data High Impedance (Maximum)	t_{CHAZx}		–	100	–	80	–	70	–	60	ns
⑧	Clock "High" to Address FC Invalid (Minimum)	t_{CHAZn}		0	–	0	–	0	–	0	–	ns
⑨[1]	Clock "High" to $\overline{AS}$, $\overline{DS}$ "Low" (Maximum)	t_{CHSLx}		–	70	–	60	–	55		55	ns
⑩	Clock "High" to $\overline{AS}$, $\overline{DS}$ "Low" (Minimum)	t_{CHSLn}		0	–	0	–	0	–	0	–	ns
⑪[2]	Address to $\overline{AS}$, $\overline{DS}$ (Read) "Low" $\overline{AS}$ Write	t_{AVSL}		35	–	30	–	20	–	0	–	ns
⑪A[2]	FC Valid to $\overline{AS}$, $\overline{DS}$ (Read) "Low" $\overline{AS}$ Write	t_{FCVSL}		70	–	60	–	50	–	40	–	ns
⑫[1]	Clock "Low" to $\overline{AS}$, $\overline{DS}$ "High"	t_{CLSH}			80	–	70	–	55	–	50	ns
⑬[2]	$\overline{AS}$, $\overline{DS}$ "High" to Address FC Invalid	t_{SHAZ}		40	–	30	–	20		10	–	ns
⑭[3]	$\overline{AS}$, $\overline{DS}$ Width "Low" (Read) $\overline{AS}$ Write	t_{SL}		337	–	240	–	195	–	160		ns
⑭A[2]	$\overline{DS}$ Width "Low" (Write)			170	–	115	–	95	–	80	–	ns
⑮[2]	$\overline{AS}$, $\overline{DS}$ Width "High"	t_{SH}		180	–	150	–	105	–	65	–	ns
⑯	Clock "High" to $\overline{AS}$, $\overline{DS}$ High Impedance	t_{CHSZ}		–	100	–	80	–	70	–	60	ns
⑰[2]	$\overline{AS}$, $\overline{DS}$ "High" to R $\overline{W}$ "High"	t_{SHRH}		50	–	40	–	20	–	10	–	ns
⑱[1]	Clock "High" to R $\overline{W}$ "High" (Maximum)	t_{CHRHx}		–	80	–	70	–	60	–	60	ns
⑱	Clock "High" to R $\overline{W}$ "High" (Minimum)	t_{CHRHn}		0	–	0		0		0	–	ns
⑳[1]	Clock "High" to R $\overline{W}$ "Low"	t_{CHRL}		–	80	–	70	–	60	–	50	ns
20A	$\overline{AS}$ "Low" to R $\overline{W}$ Valid	t_{ASRV}	Fig. 2 ~ Fig.	–	20	–	20	–	20	–	20	ns
㉑[2]	Address Valid to R $\overline{W}$ "Low"	t_{AVRL}		25	–	20	–	0	–	0	–	ns
㉑A[2]	FC Valid to R $\overline{W}$ "Low"	t_{FCVRL}		70	–	60	–	50	–	30	–	ns
㉒[2]	R $\overline{W}$ "Low" to $\overline{DS}$ "Low" (Write)	t_{RLSL}		140	–	80	–	50	–	30	–	ns
㉓	Clock "Low" to Data Out Valid	t_{CLDO}		–	80	–	70	–	55	–	55	ns
㉕[2]	$\overline{DS}$ "High" to Data Out Invalid	t_{SHDO}		40	–	30	–	20	–	15	–	ns
㉖[2]	Data Out Valid to $\overline{DS}$ "Low" (Write)	t_{DOSL}		35	–	30	–	20	–	15	–	ns
㉗[5]	Data In to Clock "Low" (Setup Time)	t_{DICL}		25	–	15	–	10	–	10	–	ns
㉘[2]	$\overline{AS}$, $\overline{DS}$ "High" to $\overline{DTACK}$ "High"	t_{SHDAH}		0	325	0	245	0	190	0	150	ns
㉙	$\overline{DS}$ "High" to Data Invalid (Hold Time)	t_{SHDI}		0	–	0	–	0	–	0	–	ns
㉚	$\overline{AS}$, $\overline{DS}$ "High" to $\overline{BERR}$ "High"	t_{SHBEH}		0	–	0	–	0	–	0	–	ns
㉛[2,5]	$\overline{DTACK}$ "Low" to Data In (Setup Time)	t_{DALDI}		–	120	–	90	–	65	–	50	ns
㉜	HALT and $\overline{RES}$ Input Transition Time	t_{RHrf}		0	200	0	200	0	200	0	200	ns
㉝	Clock "High" to $\overline{BG}$ "Low"	t_{CHGL}		–	80	–	70	–	60	–	50	ns
㉞	Clock "High" to $\overline{BG}$ "High"	t_{CHGH}			80	–	70	–	60	–	50	ns
㉟	$\overline{BR}$ "Low" to $\overline{BG}$ "Low"	t_{BRLGL}		1.5	3.5	1.5	3.5	1.5	3.5	1.5	3.5	Clk Per
㊱	$\overline{BR}$ "High" to $\overline{BG}$ "High"	t_{BRHGH}		1.5	3.5	1.5	3.5	1.5	3.5	1.5	3.5	Clk Per
㊲	$\overline{BGACK}$ "Low" to $\overline{BG}$ "High"	t_{GALGH}		1.5	3.0	1.5	3.0	1.5	3.0	1.5	3.0	Clk Per
㊲A	$\overline{BGACK}$ "Low" to $\overline{BR}$ "High"	t_{BGKBR}		20	–	20	–	20	–	20	–	ns
㊳	$\overline{BG}$ "Low" to Bus High Impedance (With $\overline{AS}$ "High")	t_{GLZ}		–	100	–	80	–	70	–	60	ns
㊴	$\overline{BG}$ Width "High"	t_{GH}		1.5	–	1.5	–	1.5	–	1.5	–	Clk Per
㊵	$\overline{BGACK}$ Width "Low"	t_{BGL}		1.5	–	1.5	–	1.5	–	1.5	–	Clk Per
㊶[5]	Asynchronous Input Setup Time	t_{ASI}		25	–	20	–	20	–	20	–	ns
㊷	$\overline{BERR}$ "Low" to $\overline{DTACK}$ "Low" (Note 3)	t_{BELDAL}		25	–	20	–	20	–	20	–	ns
㊸	Data Hold from Clock "High"	t_{CHDO}		0	–	0	–	0	–	0	–	ns
㊹	R $\overline{W}$ to Data Bus Impedance Change	t_{RLDO}		35	–	30	–	20	–	10	–	ns
㊺	HALT $\overline{RES}$ Pulse Width (Note 4)	t_{HRPW}		10	–	10	–	10	–	10	–	Clk Per

(to be continued)

Number	Item	Symbol	Test Condition	6MHz Version HD68000-6 HD68000Y6 HD68000P6 HD68000PS6		8MHz Version HD68000-8 HD68000Y8 HD68000P8 HD68000PS8		10MHz Version HD68000-10 HD68000Y10		12.5MHz Version HD68000-12 HD68000Y12		Unit
				min	max	min	max	min	max	min	max	
24	Clock "High" to R/$\overline{W}$, $\overline{VMA}$ High Impedance	t_{CHRZ}	Fig. 6, Fig. 7	-	100	–	80		70		60	ns
40	Clock "Low" to $\overline{VMA}$ "Low"	t_{CLVML}		–	80	–	70	–	70	–	70	ns
41	Clock "Low" to E Transition	t_{CLE}			85		70		55		45	ns
42	E Output Rise and Fall Time	t_{Erf}			25	–	25		25	–	25	ns
43	$\overline{VMA}$ "Low" to E "High"	t_{VMLEH}		240		200	–	150		90		ns
44	$\overline{AS}$, $\overline{DS}$ "High" to $\overline{VPA}$ "High"	t_{SHVPH}		0	160	0	120	0	90	0	70	ns
45	E "Low" to Address $\overline{VMA}$ FC Invalid	t_{ELAI}		35	-	30	–	10		10	-	ns
46	E "Low" to $\overline{AS}$, $\overline{DS}$ Invalid	t_{ELSI}		−80	–	−80		−80	–	−80	-	ns
47	E Width "High"	t_{EH}		600	–	450	–	350	–	280	-	ns
51	E Width "Low"	t_{EL}		900		700	–	550		440	-	ns
52	E Extended Rise Time	t_{CIEHX}		–	80	–	80	–	80	–	80	ns
54	Data Hold from E "Low" (Write)	t_{ELDOZ}		40	–	30		20		15		ns

(NOTES) 1. For a loading capacitance of less than or equal to 50 picofarads, subtract 5 nanoseconds from the values given in these columns.
2. Actual value depends on clock period.
3. If #47 is satisfied for both $\overline{DTACK}$ and $\overline{BERR}$, #48 may be 0 ns.
4. After V_{CC} has been applied for 100 ms.
5. If the asynchronous setup time (#47) requirements are satisfied, the $\overline{DTACK}$ low-to-data setup time (#31) requirement can be ignored. The data must only satisfy the data-in to clock-low setup time (#27) for the following cycle.



C_L = 130 pF (Includes all Parasitics)
R_L = 6.0 kΩ for $\overline{AS}$, A_1~A_{23}, $\overline{BG}$, D_0~D_{15}, E, FC_0~FC_2, $\overline{LDS}$, R/$\overline{W}$, $\overline{UDS}$, $\overline{VMA}$
*R = 1.22 kΩ for A_1~A_{23}, $\overline{BG}$, FC_0~FC_2

Figure 1 Test Loads

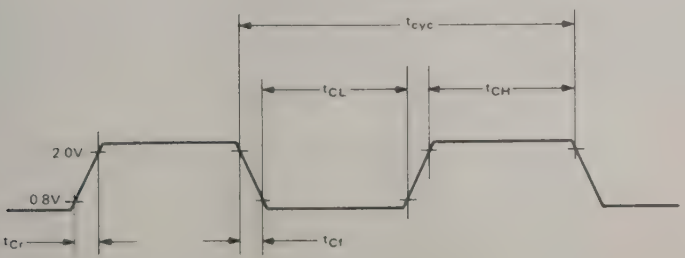
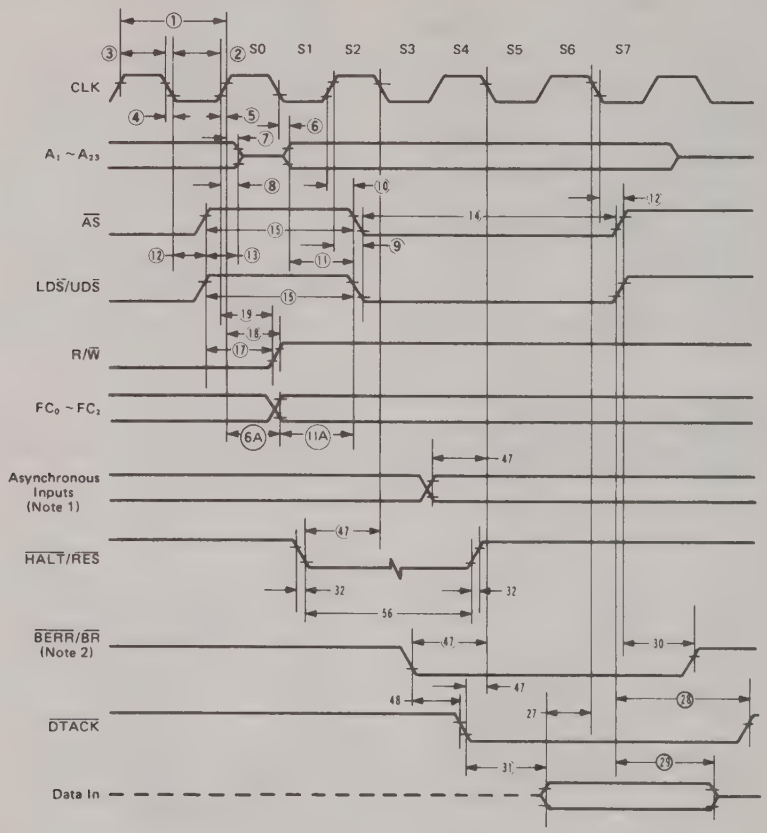
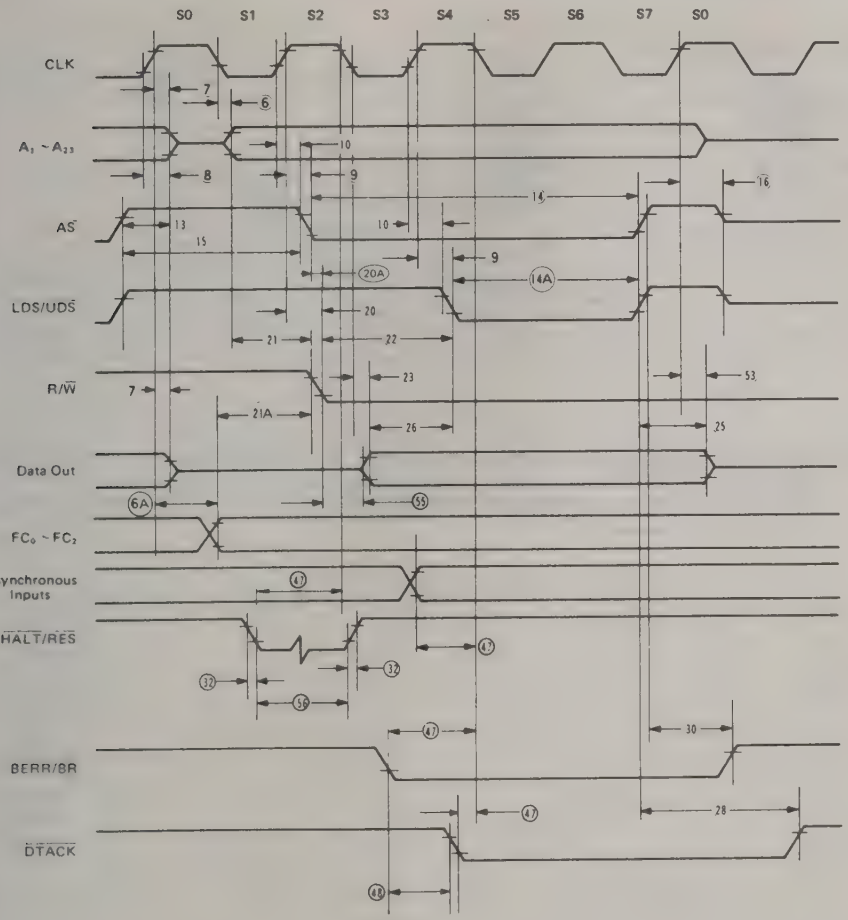


Figure 2 Input Clock Waveform



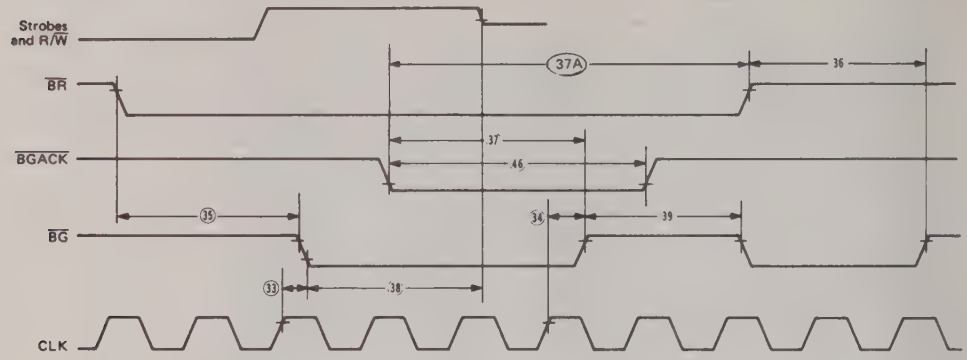
(NOTES) 1. Setup time for the asynchronous inputs $\overline{BGACK}$, $\overline{IPL_0} \sim \overline{IPL_2}$ and $\overline{VPA}$ guarantees their recognition at the next falling edge of the clock
2. $\overline{BR}$ need fall at this time only in order to insure being recognized at the end of this bus cycle.
3. Timing measurements are referenced to and from a low voltage of 0.8 volts and a high voltage of 2.0 volts, unless otherwise noted.
4. These waveforms should only be referenced in regard to the edge-to-edge measurement of the timing specifications. They are not intended as a functional description of the input and output signals. Refer to other functional descriptions and their related diagrams for device operation.

Figure 3 Read Cycle Timing



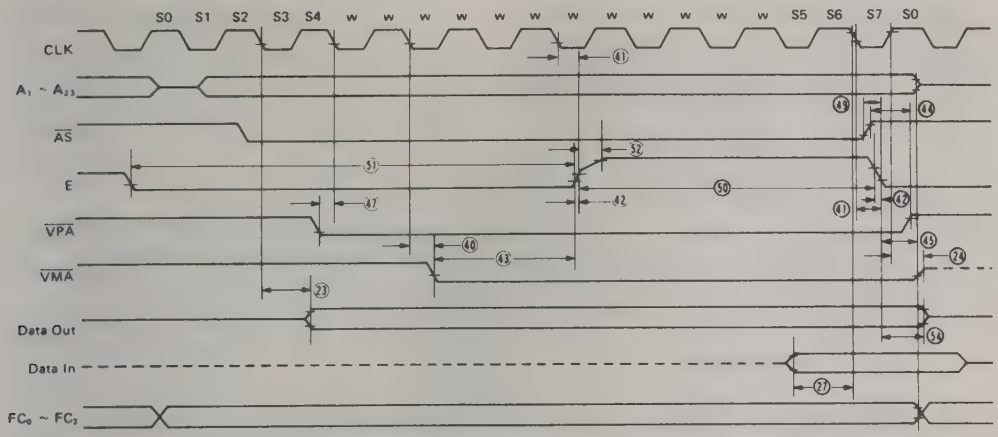
(NOTES) 1 Timing measurements are referenced to and from a low voltage of 0.8 volts and a high voltage of 2.0 volts, unless otherwise noted.
2 Because of loading variations, R/$\overline{W}$ may be valid after $\overline{AS}$ even though both are initiated by the rising edge of S2 (Specification (20A)).
3 These waveforms should only be referenced in regard to the edge-to-edge measurement of the timing specifications. They are not intended as a functional description of the input and output signals. Refer to other functional descriptions and their related diagrams for device operation.

Figure 4 Write Cycle Timing



(NOTES) 1. Setup time for the asynchronous inputs $\overline{BERR}$, $\overline{BGACK}$, $\overline{BR}$, $\overline{DTACK}$, $\overline{IPL_0} \sim \overline{IPL_2}$, and $\overline{VPA}$ guarantees their recognition at the next falling edge of the clock.
2. Waveform measurements for all inputs and outputs are specified at: logic high = 2.0 volts, logic low = 0.8 volts.
3. These waveforms should only be referenced in regard to the edge-to-edge measurement of the timing specifications. They are not intended as a functional description of the input and output signals. Refer to other functional descriptions and their related diagrams for device operation.

Figure 5 AC Electrical Waveforms — Bus Arbitration



(NOTE) This figure represents the best case HMCS6800 timing where $\overline{VPA}$ falls before the third system clock cycle after the falling edge of E.

Figure 6 HMCS6800 Timing – Best Case

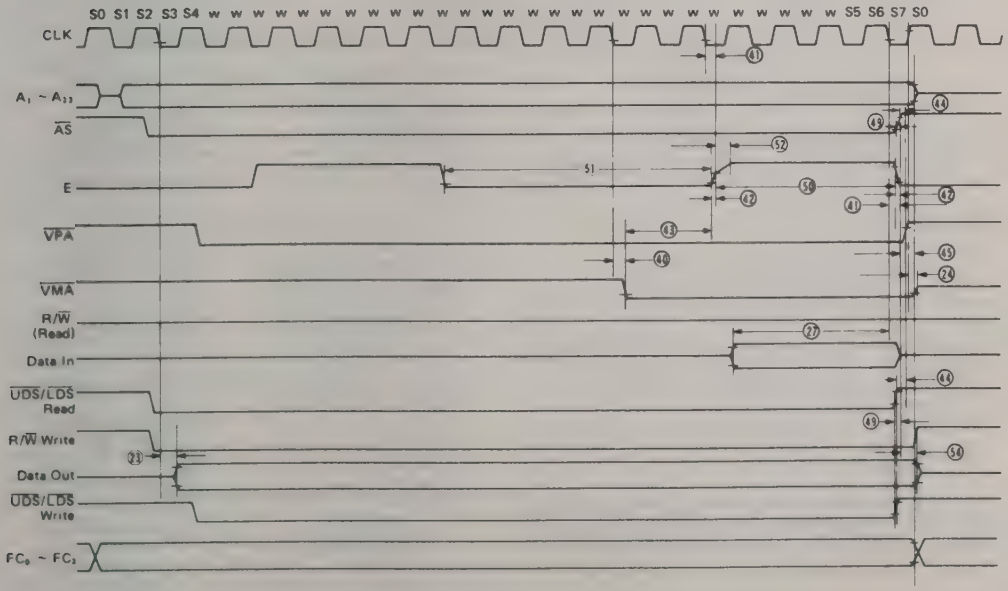

S0 S1 S2 S3 S4 W W W W W W W W W W W W W W W W W W W W W W W W W W W S5 S6 S7 S0
CLK
A1 ~ A23
AS
E
VPA
VMA
R/W (Read)
Data In
UDS/LDS Read
R/W Write
Data Out
UDS/LDS Write
FC0 ~ FC2

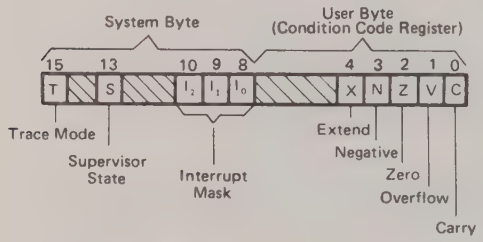
Figure 7 HMCS6800 Timing – Worst Case

■ INTRODUCTION

As shown in the programming model, the HD68000 offers seventeen 32-bit registers in addition to the 32-bit program counter and a 16-bit status register. The first eight registers (D0 ~ D7) are used as data registers for byte (8-bit), word (16-bit), and long word (32-bit) data operations. The second set of seven registers (A0 ~ A6) and the system stack pointer may be used as software stack pointers and base address registers. In addition, these registers may be used for word and long word address operations. All 17 registers may be used as index registers.

The status register contains the interrupt mask (eight levels available) as well as the condition codes; extend (X), negative (N), zero (Z), overflow (V), and carry (C). Additional status bits indicate that the processor is in a trace (T) mode and/or in a supervisor (S) state.

Status Register



Unused, read as zero.

● DATA TYPES AND ADDRESSING MODES

Five basic data types are supported. These data types are:

(1) Bits
(2) BCD Digits (4 bits)
(3) Bytes (8 bits)
(4) Word (16 bits)
(5) Long Words (32 bits)

In addition, operations on other data types such as memory address, status word data, etc., are provided for in the instruction set.

The 14 addressing modes, shown in Table 1, includes six basic types:

(1) Register Direct
(2) Register Indirect
(3) Absolute
(4) Immediate
(5) Program Counter Relative
(6) Implied

Included in the register indirect addressing modes is the capability to do postincrementing, predecrementing, offsetting and indexing. Program counter relative mode can also be modified via indexing and offsetting.

Programming Model

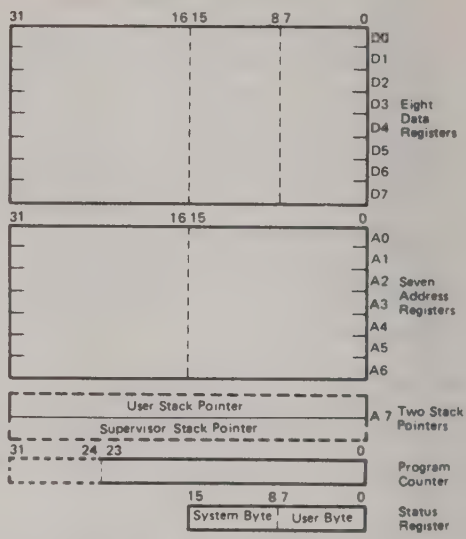


Table 1 Addressing Modes

Mode	Generation
Register Direct Addressing	
Data Register Diredt	EA = Dn
Address Register Direct	EA = An
Absolute Data Addressing	
Absolute Short	EA = (Next Word)
Absolute Long	EA = (Next Two Words)
Program Counter Relative Addressing	
Relative with Offset	EA = (PC) + d_{16}
Relative with Index and Offset	EA = (PC) + (Xn) + d_8
Register Indirect Addressing	
Register Indirect	EA = (An)
Postincrement Register Indirect	EA = (AN), An ← An + N
Predecrement Register Indirect	An ← An – N, EA = (An)
Register Indirect with Offset	EA = (An) + d_{16}
Indexed Register Indirect with Offset	EA = (An) + (Xn) + d_8
Immediate Data Addressing	
Immediate	DATA = Next Word(s)
Quick Immediate	Inherent Data
Implied Addressing	
Implied Register	EA = SR, USP, SP, PC

(NOTES)

EA = Effective Address
An = Address Register
Dn = Data Register
Xn = Address or Data Register used as Index Register
SR = Status Register
PC = Program Counter
( ) = Contents of
d_8 = Eight-bit Offset (displacement)
d_{16} = Sixteen-bit Offset (displacement)
N = 1 for Byte, 2 for Words and 4 for Long Words. If An is the stack pointer and the operand size is byte, N=2 to keep the stack pointer on a word boundary.
← = Replaces

• INSTRUCTION SET OVERVIEW

The HD68000 instruction set is shown in Table 2. Some additional instructions are variations, or subsets, of these and they appear in Table 3. Special emphasis has been given to the instruction set's support of structured high-level languages to facilitate ease of programming. Each instruction, with few exceptions, operates on bytes, words, and long words and most instructions can use any of the 14 addressing modes. Combining instruction types, data types, and addressing modes, over 1000 useful instructions are provided. These instructions include signed and unsigned multiply and divide, "quick" arithmetic operations, BCD arithmetic and expanded operations (through traps).

Table 2 Instruction Set

Mnemonic	Description
ABCD	Add Decimal with Extend
ADD	Add
AND	Logical And
ASL	Arithmetic Shift Left
ASR	Arithmetic Shift Right
B_{CC}	Branch Conditionally
BCHG	Bit Test and Change
BCLR	Bit Test and Clear
BRA	Branch Always
BSET	Bit Test and Set
BSR	Branch to Subroutine
BTST	Bit Test
CHK	Check Register Against Bounds
CLR	Clear Operand
CMP	Compare
DB_{CC}	Test Condition, Decrement and Branch
DIVS	Signed Divide
DIVU	Unsigned Divide
EOR	Exclusive Or
EXG	Exchange Registers
EXT	Sign Extend
JMP	Jump
JSP	Jump to Subroutine
LEA	Load Effective Address
LINK	Link Stack
LSL	Logical Shift Left
LSR	Logical Shift Right
MOVE	Move
MOVEM	Move Multiple Registers
MOVEP	Move Peripheral Data
MULS	Signed Multiply
MULU	Unsigned Multiply
NBCD	Negate Decimal with Extend
NEG	Negate
NOP	No Operation
NOT	One's Complement
OR	Logical Or
PEA	Push Effective Address
RESET	Reset External Devices
ROL	Rotate Left without Extend
ROR	Rotate Right without Extend
ROXL	Rotate Left with Extend
ROXR	Rotate Right with Extend
RTE	Return from Exception
RTR	Return and Restore
RTS	Return from Subroutine
SBCD	Subtract Decimal with Extend
S_{CC}	Set Conditional
STOP	Stop
SUB	Subtract
SWAP	Swap Data Register Halves
TAS	Test and Set Operand
TRAP	Trap
TRAPV	Trap on Overflow
TST	Test
UNLK	Unlink

Table 3 Variations of Instruction Types

Instruction Type	Variation	Description
ADD	ADD	Add
	ADDA	Add Address
	ADDQ	Add Quick
	ADDI	Add Immediate
	ADDX	Add with Extend
AND	AND	Logical And
	ANDI	And Immediate
	ANDI to CCR	And Immediate to Condition Codes
	ANDI to SR	And Immediate to Status Register
CMP	CMP	Compare
	CMPA	Compare Address
	CMPM	Compare Memory
	CMPI	Compare Immediate
EOR	EOR	Exclusive Or
	EORI	Exclusive Or Immediate
	EORI to CCR	Exclusive Or Immediate to Condition Codes
	EORI to SR	Exclusive Or Immediate to Status Register
MOVE	MOVE	Move
	MOVEA	Move Address
	MOVEQ	Move Quick
	MOVE from SR	Move from Status Register
	MOVE to SR	Move to Status Register
	MOVE to CCR	Move to Condition Codes
	MOVE USP	Move User Stack Pointer
NEG	NEG	Negate
	NEGX	Negate with Extend
OR	OR	Logical Or
	ORI	Or Immediate
	ORI to CCR	Or Immediate to Condition Codes
	ORI to SR	Or Immediate to Status Register
SUB	SUB	Subtract
	SUBA	Subtract Address
	SUBI	Subtract Immediate
	SUBQ	Subtract Quick
	SUBX	Subtract with Extend

■ REGISTER DESCRIPTION AND DATA ORGANIZATION

The following paragraphs describe the registers and data organization of the HD68000.

• OPERAND SIZE

Operand sizes are defined as follows: a byte equals 8 bits, a word equals 16 bits, and a long word equals 32 bits. The operand size for each instruction is either explicitly encoded in the instruction or implicitly defined by the instruction operation. Implict instructions support some subset of all three sizes.

• DATA ORGANIZATION IN REGISTERS

The eight data registers support data operands of 1, 8, 16, or 32 bits. The seven address registers together with the active stack pointer support address operands of 32 bits.

DATA REGISTERS

Each data register is 32 bits wide. Byte operands occupy the low order 8 bits, word operands the low order 16 bits, and long word operands the entire 32 bits. The least significant bit is addressed as bit zero; the most significant bit is addressed as bit 31.

When a data register is used as either a source or destination operand, only the appropriate low-order portion is changed; the remaining high-order portion is neither used nor changed.

ADDRESS REGISTERS

Each address register and the stack pointer is 32 bits wide and holds a full 32 bit address. Address registers do not support byte sized operands. Therefore, when an address register is used as a source operand, either the low order word or the entire long word operand is used depending upon the operation size. When an address register is used as the destination operand, the entire register is affected regardless of the operation size. If the operation size is word, any other operands are sign extended to 32 bits before the operation is performed.

• DATA ORGANIZATION IN MEMORY

Bytes are individually addressable with the high order byte having an even address the same as the word, as shown in Figure 8. The low order byte has an odd address that is one count higher than the word address. Instructions and multibyte data are accessed only on word (even byte) boundaries. If a long word datum is located at address n (n even), then the second word of that datum is located at address n + 2.

The data types supported by the HD68000 are: bit data, integer data of 8, 16, or 32 bits, 32-bit addresses and binary coded decimal data. Each of these data types is put in memory, as shown in Figure 9. The numbers indicate the order in which the data would be accessed from the processor.

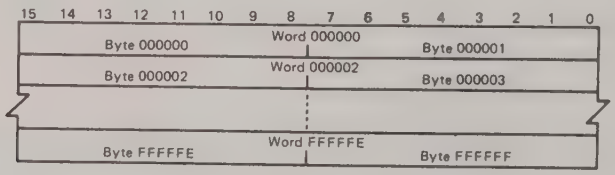


Figure 8 Word Organization in Memory

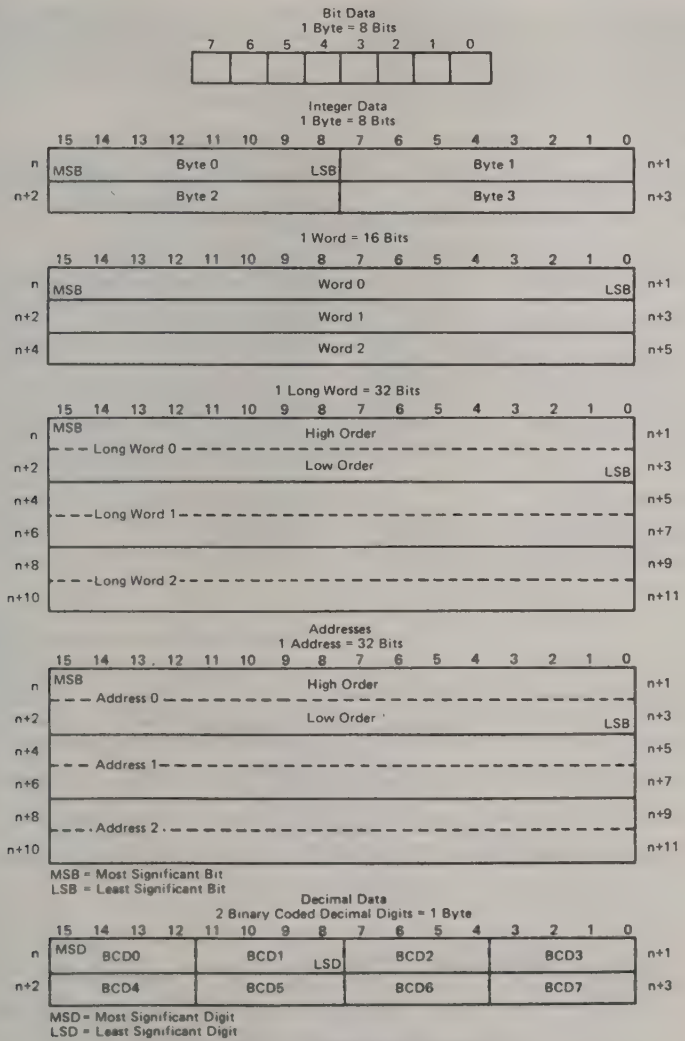


Figure 9 Data Organization in Memory

• ADDRESSING

Instructions for the HD68000 contain two kinds of information: the type of function to be performed, and the location of the operand(s) on which to perform that function. The methods used to locate (address) the operand(s) are explained in the following paragraphs.

Instructions specify an operand location in one of three ways:

Register Specification – the number of the register is given in the register field of the instruction.

Effective Address – use of the different effective address modes.

Implicit Reference – the definition of certain instructions implies the use of specific registers.

• INSTRUCTION FORMAT

Instructions are from one to five words in length, as shown in Figure 10. The length of the instruction and the operation to be performed is specified by the first word of the instruction which is called the operation word. The remaining words further specify the operands. These words are either immediate operands or extensions to the effective address mode specified in the operation word.

• PROGRAM/DATA REFERENCES

The HD68000 separates memory references into two classes: program references, and data references. Program references, as the name implies, are references to that section of memory that contains the program being executed. Data references refer to that section of memory that contains data. Operand reads are from the data space except in the case of the program counter relative addressing mode. All operand writes are to the data space.

• REGISTER SPECIFICATION

The register field within an instruction specifies the register to be used. Other fields within the instruction specify whether the register selected is an address or data register and how the register is to be used.

• EFFECTIVE ADDRESS

Most instructions specify the location of an operand by using the effective address field in the operation word. For example, Figure 11 shows the general format of the single effective address instruction operation word. The effective address is composed of two 3-bit fields: the mode field, and the register field. The value in the mode field selects the different address modes. The register field contains the number of a register.

The effective address field may require additional information to fully specify the operand. This additional information, called the effective address extension, is contained in the following word or words and is considered part of the instruction, as shown in Figure 10. The effective address modes are grouped into three categories: register direct, memory addressing, and special.

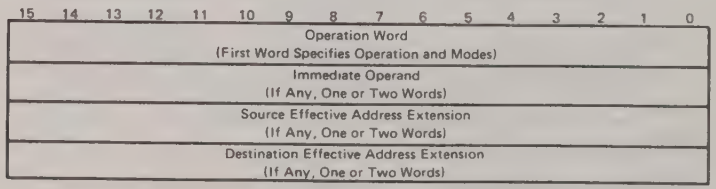

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Operation Word
(First Word Specifies Operation and Modes)
Immediate Operand
(If Any, One or Two Words)
Source Effective Address Extension
(If Any, One or Two Words)
Destination Effective Address Extension
(If Any, One or Two Words)

Figure 10 Instruction Format

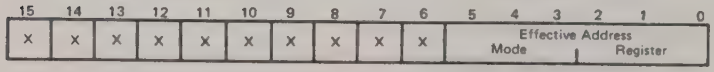


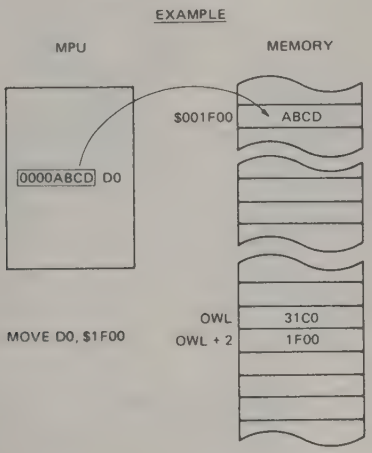
Figure 11 Single-Effective-Address Instruction Operation Word General Format

REGISTER DIRECT MODES

These effective addressing modes specify that the operand is in one of the 16 multifunction registers.

Data Register Direct

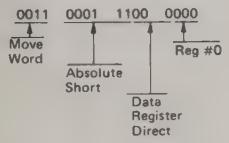
The operand is in the data register specified by the effective address register field.



COMMENTS
- EA = Dn

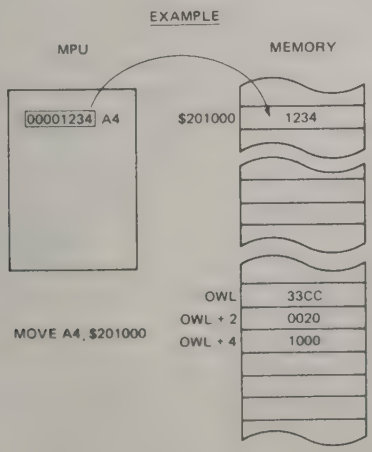
- Machine Level Coding

MOVE D0, $1F00



Address Register Direct

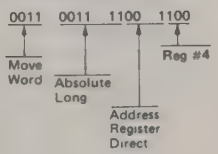
The operand is in the address register specified by the effective address register field.

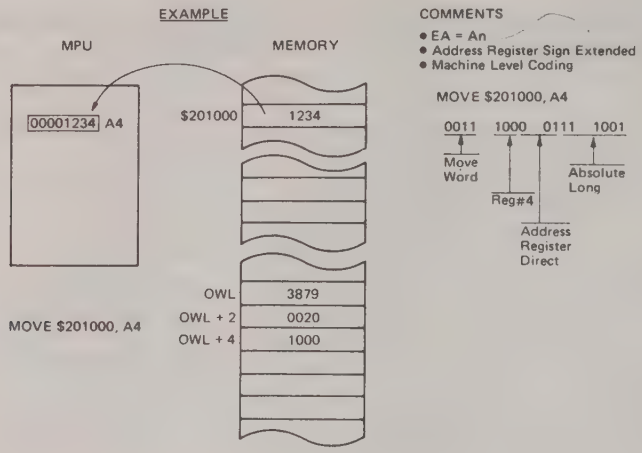


COMMENTS
- EA = An

- Machine Level Coding

MOVE A4, $201000



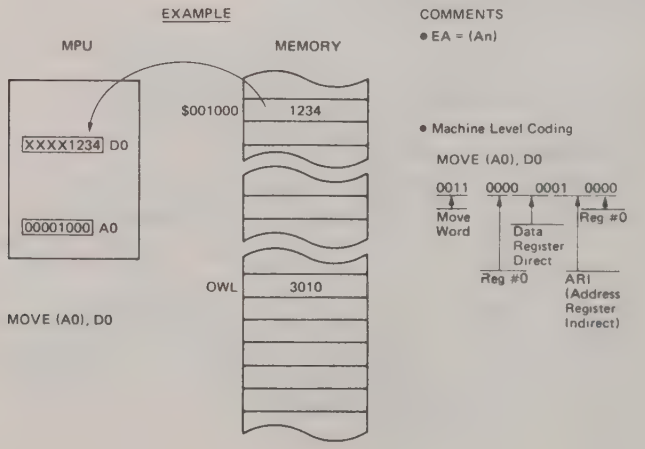


MEMORY ADDRESS MODES

These effective addressing modes specify that the operand is in memory and provide the specific address of the operand.

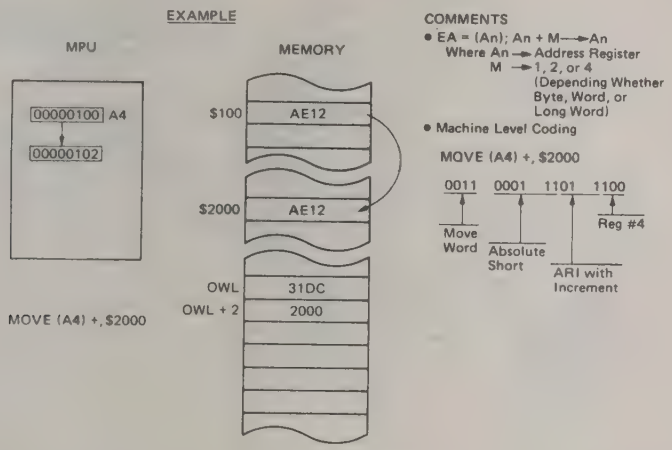
Address Register Indirect

The address of the operand is in the address register specified by the register field. The reference is classified as a data reference with the exception of the jump and jump to subroutine instructions.



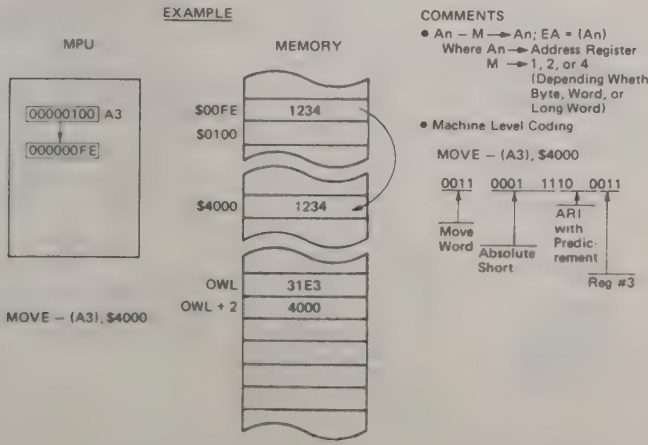
Address Register Indirect With Postincrement

The address of the operand is in the address register specified by the register field. After the operand address is used, it is incremented by one, two, or four depending upon whether the size of the operand is byte, word, or long word. If the address register is the stack pointer and the operand size is byte, the address is incremented by two rather than one to keep the stack pointer on a word boundary. The reference is classified as a data reference.



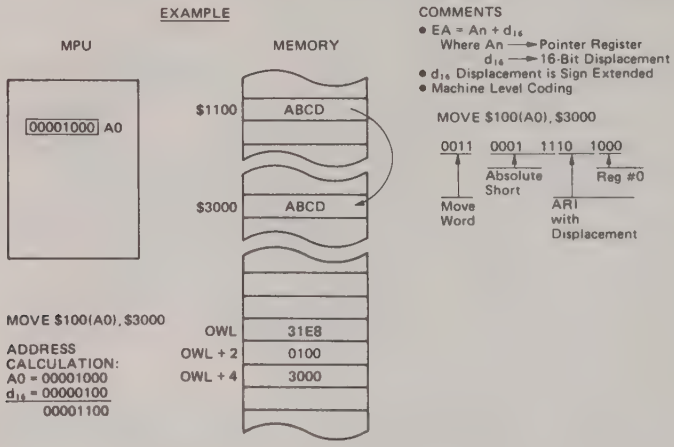
Address Register Indirect With Predecrement

The address of the operand is in the address register specified by the register field. Before the operand address is used, it is decremented by one, two, or four depending upon whether the operand size is byte, word, or long word. If the address register is the stack pointer and the operand size is byte, the address is decremented by two rather than one to keep the stack pointer on a word boundary. The reference is classified as a data reference.



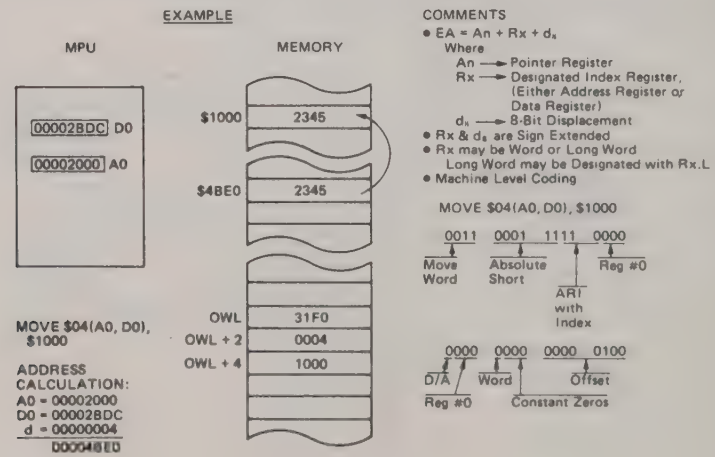
Address Register Indirect With Displacement

This address mode requires one word of extension. The address of the operand is the sum of the address in the address register and the sign-extended 16-bit displacement integer in the extension word. The reference is classified as a data reference with the exception of the jump to subroutine instructions.



Address Register Indirect With Index

This address mode requires one word of extension. The address of the operand is the sum of the address in the address register, the sign-extended displacement integer in the low order eight bits of the extension word, and the contents of the index register. The reference is classified as a data reference with the exception of the jump and jump to subroutine instructions.

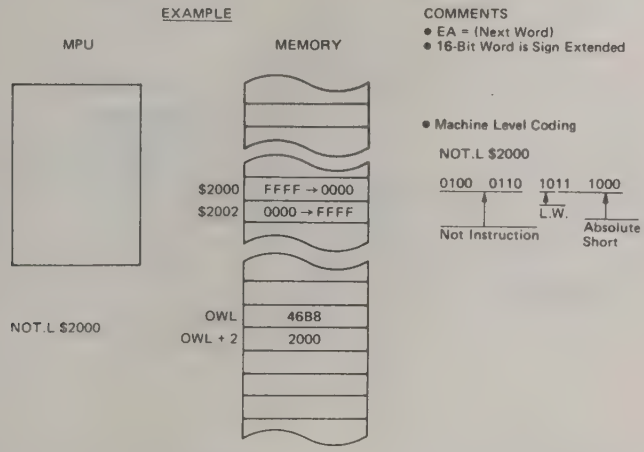


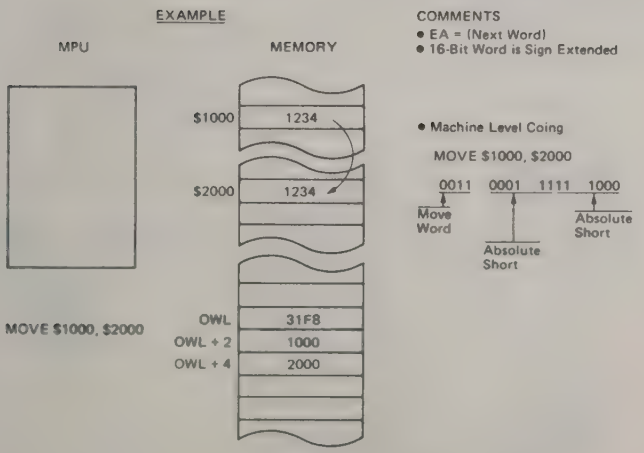
SPECIAL ADDRESS MODE

The special address modes use the effective address register field to specify the special addressing mode instead of a register number.

Absolute Short Address

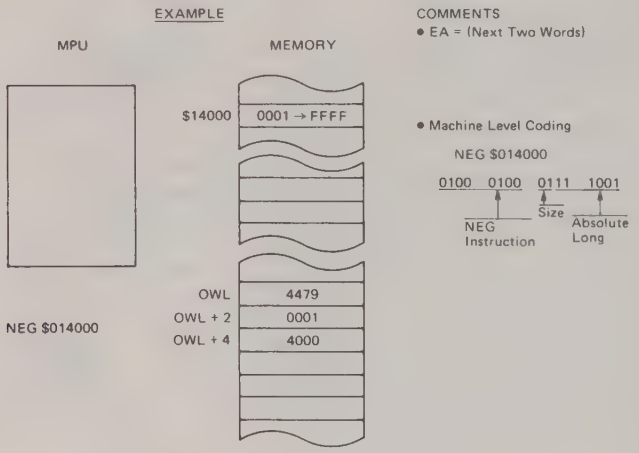
This address mode requires one word of extension. The address of the operand is the extension word. The 16-bit address is sign extended before it is used. The reference is classified as a data reference with the exception of the jump and jump to subroutine instructions.





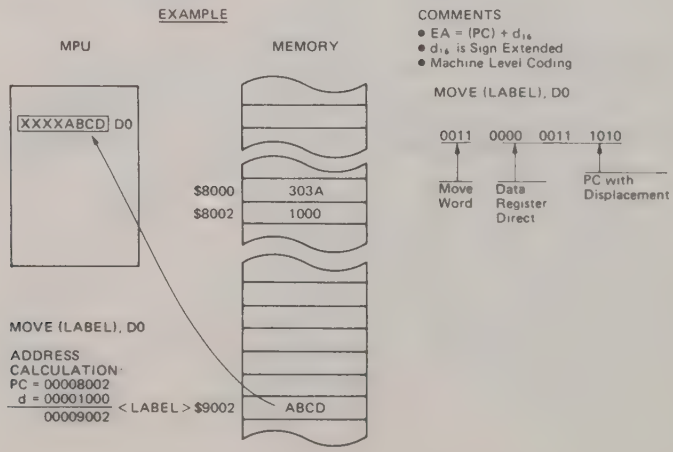
Absolute Long Address

This address mode requires two words of extension. The address of the operand is developed by the concatenation of the extension words. The high-order part of the address is the first extension word; the low-order part of the address is the second extension word. The reference is classified as a data reference with the exception of the jump and jump to subroutine instructions.



Program Counter With Displacement

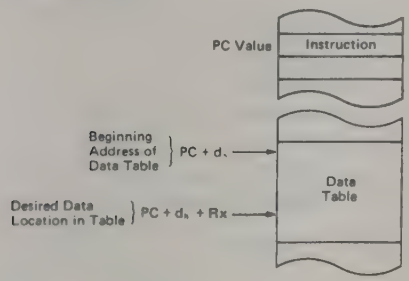
This address mode requires one word of extension. The address of the operand is the sum of the address in the program counter and the sign-extended 16-bit displacement integer in the extension word. The value in the program counter is the address of the extension word. The reference is classified as a program reference.



Program Counter With Index

This address mode requires one word of extension. This address is the sum of the address in the program counter, the sign-extended displacement integer in the lower eight bits of the extension word, and the contents of the index register. The value in the program counter is the address of the extension word. This reference is classified as a program reference.

EA = (PC) + (Rx) + d_s



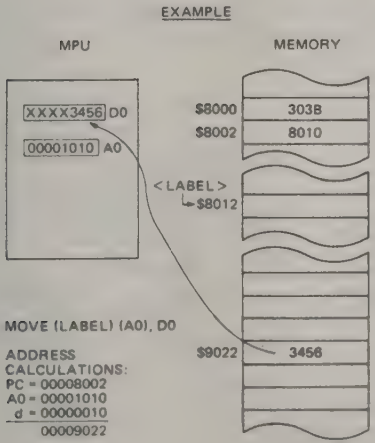
(NOTE)

Extension Word

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D/A	Register			W/L	0	0	0	Displacement Integer							

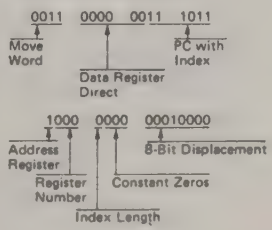
D/A : Data Register = 0, Address Register = 1
Register : Index Register Number
W/L : Sign-extented, low order Word integer in Index Register = 0
Long Word in Index Register = 1

EXAMPLE



COMMENTS
- EA = (PC) + (Rx) + d_s
  Where
  PC —► Current Program Counter
  Rx —► Designated Index Register (Either Data or Address Register)
  d_s —► 8-Bit Displacement
- Rx and d_s are Sign Extended
- Rx may be Word or Long Word
  Long Word is Designated with Rx.L
- Machine Level Coding

MOVE (LABEL) (A0), D0



Immediate Data

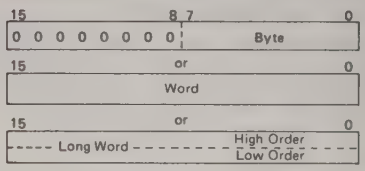
This address mode requires either one or two words of extension depending on the size of the operation.

Byte operation – operand is low order byte of extension word

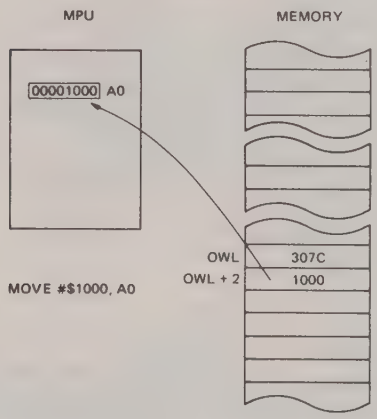
Word operation – operand is extension word

Long word operation – operand is in the two extension words, high-order 16 bits are in the first extension word, low-order 16 bits are in the second extension word.

Extension Word



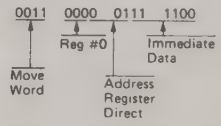
EXAMPLE



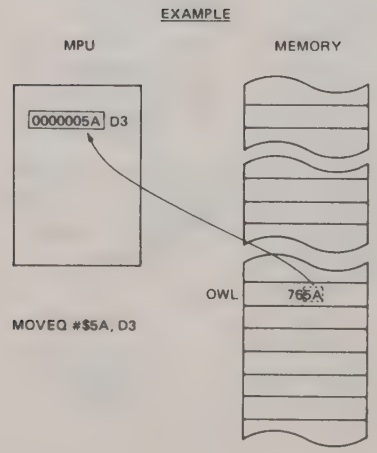
COMMENTS

- Data = Next Word(s)
- Data is Sign Extended for Address Register but not Data Register
- Machine Level Coding

MOVE #$1000, A0



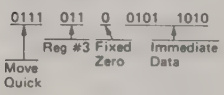
EXAMPLE



COMMENTS

- Inherent Data
- Data is Sign Extended to Long Word
- Destination must be a Data Register
- Machine Level Coding

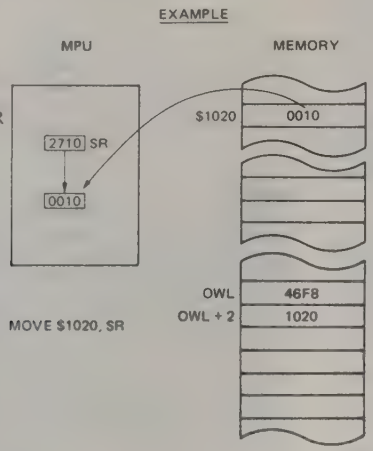
MOVEQ #$5A, D3



Condition Codes or Status Register

A selected set of instructions may reference the status register by means of the effective address field. These are:

ANDI to CCR
ANDI to SR
EORI to CCR
EORI to SR
ORI to CCR
ORI to SR
MOVE to CCR
MOVE to SR
MOVE from SR



COMMENTS
- EA = (Next Word)
- Note: This Example is a Privileged Instruction
- Machine Level Coding

MOVE $1020, SR

0100 0110 1111 1000

Move to SR — Absolute Short

• EFFECTIVE ADDRESS ENCODING SUMMARY

Table 4 is a summary of the effective addressing modes discussed in the previous paragraphs.

Table 4 Effective Address Encoding Summary

Addressing Mode	Mode	Register
Data Register Direct	000	register number
Address Register Direct	001	register number
Address Register Indirect	010	register number
Address Register Indirect with Postincrement	011	register number
Address Register Indirect with Predecrement	100	register number
Address Register Indirect with Displacement	101	register number
Address Register Indirect with Index	110	register number
Absolute Short	111	000
Absolute Long	111	001
Program Counter with Displacement	111	010
Program Counter with Index	111	011
Immediate	111	100

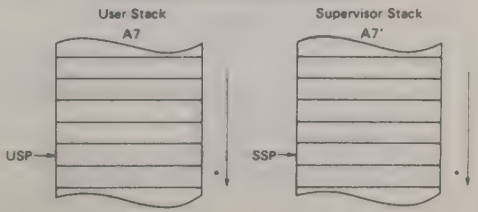
• IMPLICIT REFERENCE

Some instructions make implicit reference to the program counter (PC), the system stack pointer (SP), the supervisor stack pointer (SSP), the user stack pointer (USP), or the status register (SR).

SYSTEM STACK

The system stack is used implicitly by many instructions; user stacks and queues may be created and maintained through the addressing modes. Address register seven (A7) is the system stack pointer (SP). The system stack pointer is either the supervisor stack pointer (SSP) or the user stack pointer (USP), depending on the state of the S-bit in the status register. If the S-bit indicates supervisor state, SSP is the active system stack pointer, and the USP cannot be referenced as an address register. If the S-bit indicates user state, the USP is the active system stack pointer, and the SSP cannot be referenced. Each system stack fills from high memory to low memory.

SYSTEM STACK POINTERS



- Accessed when S = 0
- PC is Stacked on Subroutine Calls in User State
- Increasing Addresses

- Accessed when S = 1
- PC is Stacked on Subroutine Calls in Supervisor State
- Used for Exception Processing

The address mode SP@– creates a new item on the active system stack, and the address mode SP@+ deletes an item from the active system stack.

The program counter is saved on the active system stack on subroutine calls, and restored from the active system stack on returns. On the other hand, both the program counter and the status register are saved on the supervisor stack during the processing of traps and interrupts. Thus, the correct execution of the supervisor state code is not dependent on the behavior of user code and user programs may use the user stack pointer arbitrarily.

In order to keep data on the system stack aligned properly, data entry on the stack is restricted so that data is always put in the stack on a word boundary. Thus byte data is pushed on or pulled from the system stack in the high order half of the word; the lower half is unchanged.

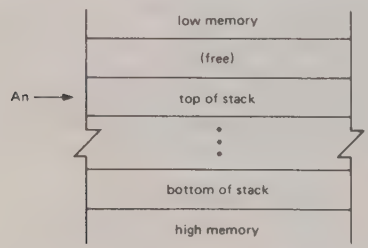
USER STACKS

User stacks can be implemented and manipulated by employing the address register indirect with postincrement and predecrement addressing modes. Using an address register (on of A0 through A6), the user may implement stacks which are filled either from high memory to low memory, or vice versa. The important things to remember are:

- using predecrement, the register is decremented before its contents are used as the pointer into the stack,
- using postincrement, the register is incremented after its contents are used as the pointer into the stack,
- byte data must be put on the stack in pairs when mixed with word or long data so that the stack will not get misaligned when the data is retrieved. Word and long accesses must be on word boundary (even) addresses.

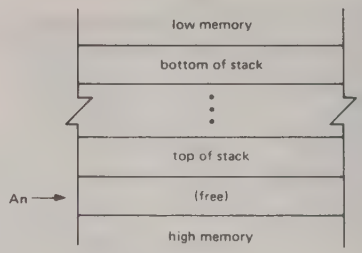
Stack growth from high to low memory is implemented with
An@– to push data on the stack,
An@+ to pull data from the stack.

After eigher a push or a pull operation, register An points to the last (top) item on the stack. This is illustrated as:



Stack growth from low to high memory is implemented with
An@+ to push data on the stack,
An@– to pull data from the stack.

After either a push or a pull operation, register An points to the next available space on the stack. This is illustrated as:

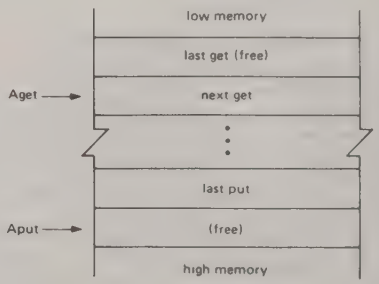


QUEUES

User queues can be implemented and manipulated with the address register indirect with postincrement or predecrement addressing modes. Using a pair of address registers (two of A0 through A6), the user may implement queues which are filled either from high memory to low memory, or vice versa. Because queues are pushed from one end and pulled from the other, two registers are used: the put and get pointers.

Queue growth from low to high memory is implemented with
Aput@+ to put data into the queue,
Aget@+ to get data from the queue.

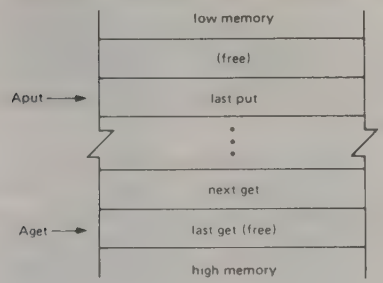
After a put operation, the put address register points to the next available space in the queue and the unchanged get address register points to the next item to remove from the queue. After a get operation, the get address register points to the next item to remove from the queue and the unchanged put address register points to the next available space in the queue. This is illustrated as:



If the queue is to be implemented as a circular buffer, the address register should be checked and, if necessary, adjusted before the put or get operation is performed. The address register is adjusted by subtracting the buffer length (in bytes).

Queue growth from high to low memory is implemented with
Aput@– to put data into the queue,
Aget@– to get data from the queue.

After a put operation, the put address register points to the last item put in the queue, and the unchanged get address register points to the last item removed from the queue. After a get operation, the get address register points to the last item removed from the queue and the unchanged put address register points to the last item put in the queue. This is illustrated as:



If the queue is to be implemented as a circular buffer, the get or put operation should be performed first, and then the address register should be checked and, if necessary, adjusted. The address register is adjusted by adding the buffer length (in bytes).

■INSTRUCTION SET SUMMARY

The following paragraphs contain an overview of the form and structure of the HD68000 instruction set. The instructions form a set of tools that include all the machine functions to perform the following operations:

Data Movement
Integer Arithmetic
Logical
Shift and Rotate
Bit Manipulation
Binary Coded Decimal
Program Control
System Control

The complete range of instruction capabilities combined with the flexible addressing modes described previously provide a very flexible base for program development.

• DATA MOVEMENT OPERATIONS

The basic method of data acquisition (transfer and storage) is provided by the move (MOVE) instruction. The move instruction and the effective addressing modes allow both address and data manipulation. Data move instructions allow byte, word, and long word operands to be transferred from memory to memory, memory to register, register to memory, and register to memory, and register to register. Address move instructions allow word and long word operand transfers and ensure that only legal address manipulations are executed. In addition to the general move instruction there are several special data movement instructions: move multiple registers (MOVEM), move peripheral data (MOVEP), exchange registers (EXG), load effective address (LEA), push effective address (PEA), link stack (LINK), unlink stack (UNLK), and move quick (MOVEQ). Table 5 is a summary of the data movement operations.

Table 5 Data Movement Operations

Instruction	Operand Size	Operation
EXG	32	Rx ↔ Ry
LEA	32	EA → An
LINK	–	An → – (SP) SP → An; SP + d → SP
MOVE	8, 16, 32	(EA)s → EAd
MOVEM	16, 32	(EA) → An, Dn An, Dn → EA
MOVEP	16, 32	(EA) → Dn Dn → EA
MOVEQ	8	#xxx → Dn
PEA	32	EA → –(SP)
SWAP	32	Dn[31:16] ↔ Dn[15:0]
UNLK	–	An → Sp; (SP) + → An

(NOTES)
s = source
d = destination
[ ] = bit numbers
–( ) = indirect with predecrement
( ) + = indirect with postincrement
= immediate data

• INTEGER ARITHMETIC OPERATIONS

The arithmetic operations include the four basic operations of add (ADD), subtract (SUB), multiply (MUL), and divide (DIV) as well as arithmetic compare (CMP), clear (CLR), and negate (NEG). The add and subtract instructions are available for both address and data operations, with data operations accepting all operand sizes. Address operations are limited to legal address size operands (16 or 32 bits). Data, address, and memory compare operations are also available. The clear and negate instructions may be used on all sizes of data operands.

The multiply and divide operations are available for signed and unsigned operands using word multiply to produce a long word product, and a long word dividend with word divisor to produce a word quotien with a word remainder.

Multiprecision and mixed size arithmetic can be accomplished using a set of extended instructions. These instructions are: add extended (ADDX), subtract extended (SUBX), sign extend (EXT), and negate binary with extend (NEGX).

A test operand (TST) instruction that will set the condition codes as a result of a compare of the operand with zero is also available. Test and set (TAS) is a synchronization instruction useful in multiprocessor systems. Table 6 is a summary of the integer arithmetic operations.

Table 6 Integer Arithmetic Operations

Instruction	Operand Size	Operation
ADD	8, 16, 32	Dn + (EA) → Dn (EA)+ Dn → EA (EA) + #xxx → EA
	16, 32	AN + (EA) → An
ADDX	8, 16, 32 16, 32	Dx + Dy + X → Dx –(Ax) + – (Ay) + X → (Ax)
CLR	8, 16, 32	(EA) → MPU 0 → EA
CMP	8, 16, 32	Dn - (EA) (EA) - #xxx (Ax) + – (Ay) +
	16, 32	An - (EA)
DIVS	32 ÷ 16	Dn÷(EA)→ Dn
DIVU	32 ÷ 16	Dn÷(EA)→ Dn
EXT	8 → 16 16 → 32	$(Dn)_8 \rightarrow Dn_{16}$ $(Dn)_{16} \rightarrow Dn_{32}$
MULS	16×16 → 32	Dn×(EA) → Dn
MULU	16×16 → 32	Dn×(EA) → Dn
NEG	8, 16, 32	0 - (EA) → EA
NEGX	8, 16, 32	0 - (EA) - X - EA
SUB	8, 16, 32	Dn - (EA) → Dn (EA) - Dn → EA (EA) - #xxx → EA
	16, 32	An - (EA) → An
SUBX	8, 16, 32	Dx - Dy - X → Dx –(Ax) – – (Ay) – X → (Ax)
TAS	8	(EA) - 0, 1 → EA[7]
TST	8, 16, 32	(EA) - 0

(NOTE) [ ] = bit number
– ( ) = indirect with predecrement
( ) + = indirect with postincrement
= immediate data

• LOGICAL OPERATIONS

Logical operation instructions AND, OR, EOR, and NOT are available for all sizes of integer data operands. A similar set of immediate instructions (ANDI, ORI, and EORI) provide these logical operations with all sizes of immediate data. Table 7 is a summary of the logical operations.

Table 7 Logical Operations

Instruction	Operand Size	Operation
AND	8, 16, 32	Dn∧(EA) → Dn (EA)∧ Dn → EA (EA)∧ =xxx → EA
OR	8, 16, 32	Dn ∨ (EA) → Dn (EA) ∨ Dn → EA (EA) ∨ =xxx → EA
EOR	8, 16, 32	(EA)⊕Dy → EA (EA)⊕ #xxx → EA
NOT	8, 16, 32	~ (EA) - EA

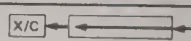
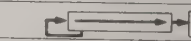
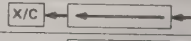
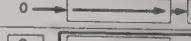
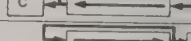
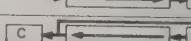
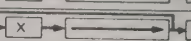
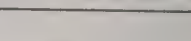
(NOTE) ~ = invert
∨ = logical OR
= immediate data
∧ = logical AND
⊕ = exclusive OR

• SHIFT AND ROTATE OPERATIONS

Shift operations in both directions are provided by the arithmetic instructions ASR and ASL and logical shift instructions LSR and LSL. The rotate instructions (with and without extend) available are ROXR, ROXL, ROR, and ROL. All shift and rotate operations can be performed in either registers or memory. Register shifts and rotates support all operand sizes and allow a shift count specified in the instruction of one to eight bits, or 0 to 63 specified in a data register.

Memory shifts and rotates are for word operands only and allow only single-bit shifts or rotates. Table 8 is a summary of the shift and rotate operations.

Table 8 Shift and Rotate Operations

Instruction	Operand Size	Operation
ASL	8, 16, 32	
ASR	8, 16, 32	
LSL	8, 16, 32	
LSR	8, 16, 32	
ROL	8, 16, 32	
ROR	8, 16, 32	
ROXL	8, 16, 32	
ROXR	8, 16, 32	

• BIT MANIPULATION OPERATIONS

Bit manipulation operations are accomplished using the following instructions: bit test (BTST), bit test and set (BSET), bit test and clear (BCLR), and bit test and change (BCHG). Table 9 is a summary of the bit manipulation operations. (Bit 2 of the status register is Z.)

Table 9 Bit Manipulation Operations

Instruction	Operand Size	Operation
BTST	8, 32	~ bit of (EA) → Z
BSET	8, 32	~ bit of (EA) → Z; 1 → bit of EA
BCLR	8, 32	~ bit of (EA) → Z; 0 → bit of EA
BCHG	8, 32	~ bit of (EA) → Z; ~ bit of (EA) → bit of EA

(Note) ~ = invert

• BINARY CODED DECIMAL OPERATIONS

Multiprecision arithmetic operations on binary coded decimal numbers are accomplished using the following instructions: add decimal with extend (ABCD), subtract decimal with extend (SBCD), and negate decimal with extend (NBCD). Table 10 is a summary of the binary coded decimal operations.

Table 10 Binary Coded Decimal Operations

Instruction	Operand Size	Operation
ABCD	8	$Dx_{10} + Dy_{10} + X \rightarrow Dx$ $-(Ax)_{10} + -(Ay)_{10} + X \rightarrow (Ax)$
SBCD	8	$Dx_{10} - Dy_{10} - X \rightarrow Dx$ $-(Ax)_{10} - -(Ay)_{10} - X \rightarrow (Ax)$
NBCD	8	$0 - (EA)_{10} - X \rightarrow EA$

– ( ) = indirect with predecrement

• PROGRAM CONTROL OPERATIONS

Program control operations are accomplished using a series of conditional and unconditional branch instructions and return instructions. These instructions are summarized in Table 11.

The conditional instructions provide setting and branching for the following conditions:

CC – carry clear	LS – low or same
CS – carry set	LT – less than
EQ – equal	MI – minus
F – never true	NE – not equal
GE – greater or equal	PL – plus
GT – greater than	T – always true
HI – high	VC – no overflow
LE – less or equal	VS – overflow

Table 11 Program Control Operations

Instruction	Operation
Conditional	
B_{CC}	Branch conditionally (14 conditions) 8- and 16-bit displacement
DB_{CC}	Test condition, decrement, and branch 16-bit displacement
S_{CC}	Set byte conditionally (16 conditions)
Unconditional	
BRA	Branch always 8-and 16-bit displacement
BSR	Branch to subroutine 8- and 16-bit displacement
JMP	Jump
JSR	Jump to subroutine
Returns	
RTR	Return and restore condition codes
RTS	Return from subroutine

• SYSTEM CONTROL OPERATIONS

System control operations are accomplished by using privileged instructions, trap generating instructions, and instructions that use or modify the status register. These instructions are summarized in Table 12.

Table 12 System Control Operations

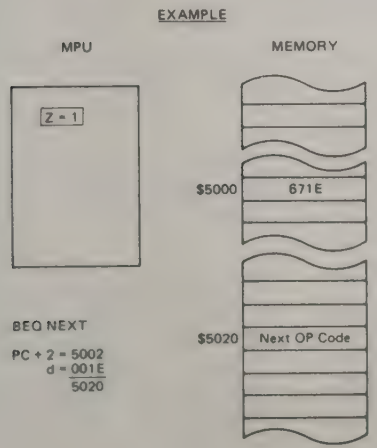
Instruction	Operation
Privileged	
RESET	Reset external devices
RTE	Return from exception
STOP	Stop program execution
ORI to SR	Logical OR to status register
MOVE USP	Move user stack pointer
ANDI to SR	Logical AND to status register
EORI to SR	Logical EOR to status register
MOVE EA to SR	Load new status register
Trap Generating	
TRAP	Trap
TRAPV	Trap on overflow
CHK	Check register against bounds
Status Register	
ANDI to CCR	Logical AND to condition codes
EORI to CCR	Logical EOR to condition codes
MOVE EA to CCR	Load new condition codes
ORI to CCR	Logical OR to condition codes
MOVE SR to EA	Store status register

• BRANCH INSTRUCTION ADDRESSING

BRANCH INSTRUCTION FORMAT

	15 – 8	7 – 0
Operation Word	Operation Code	8 bit Displacement
Extension Word	16 bit Displacement if 8 bit Displacement = 0	

RELATIVE, FORWARD REFERENCE, 8-BIT OFFSET



COMMENTS
- Offset Contained in 8 LSBs of Op Word
- Offset is 2's Complement Number
- If Offset = 0 then Word Offset is Used
- Machine Level Coding

BEQ NEXT

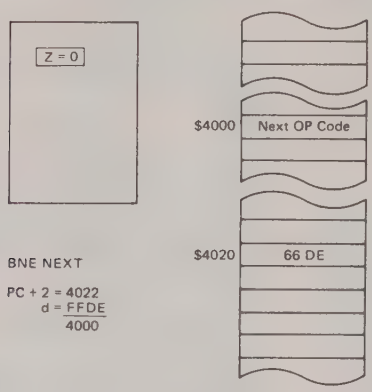
0110 0111 0001 1110

Branch — Offset

Branch If Equal

RELATIVE, BACKWARD REFERENCE 8-BIT OFFSET

EXAMPLE



COMMENTS
- Offset Contained in 8 LSBs of Op Word
- Offset is 2's Complement Number
- If Offset = 0 then Word Offset is Used
- Machine Level Coding

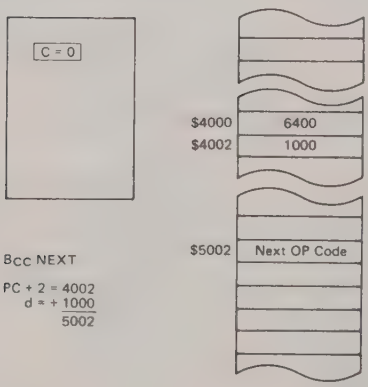
BNE NEXT

0110 0110 1101 1110

Branch — Branch If Not Equal — Offset

RELATIVE, FORWARD REFERENCE, 16-BIT OFFSET

EXAMPLE



COMMENTS
- Offset in Next Word
- 8-Bit Offset Field = 0
- 2's Complement Offset
- Machine Level Coding

BCC NEXT

0110 0100 0000 0000

Branch — Branch If Carry Clear — Zero Offset

■ SIGNAL AND BUS OPERATION DESCRIPTION

The following paragraphs contain a brief description of the input and output signals. A discussion of bus operation during the various machine cycles and operations is also given.

(NOTE) The terms **assertion** and **negation** will be used extensively. This is done to avoid confusion when dealing with a mixture of "active-low" and "active-high" signals. The term assert or assertion is used to indicate that a signal is active or true independent of whether that voltage is low or high. The term negate or negation is used to indicate that a signal is inactive or false.

• SIGNAL DESCRIPTION

The input and output signals can be functionally organized into the groups shown in Figure 12. The following paragraphs provide a brief description of the signals and also a reference (if applicable) to other paragraphs that contain more detail about the function being performed.

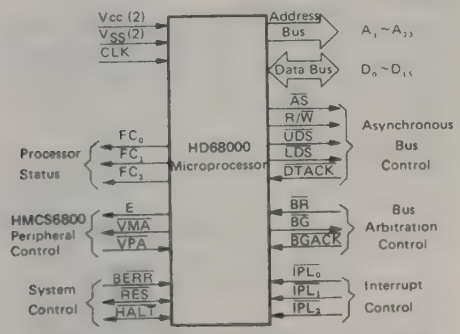


Figure 12 Input and Output Signals

ADDRESS BUS (A_1 through A_{23})

This 23-bit, unidirectional, three-state bus is capable of addressing 8 megawords of data. It provides the address for bus operation during all cycles except interrupt cycles. During interrupt cycles, address lines A_1, A_2, and A_3, Provide information about what level interrupt is being serviced while address lines A_4 through A_{23} are all set to a logic high.

DATA BUS (D_0 through D_{15})

This 16-bit, bidirectional, three-state bus is the general purpose data path. It can transfer and accept data in either word or byte length. During an interrupt acknowledge cycle, an external device supplies the vector number on data lines D_0 through D_7.

ASYNCHRONOUS BUS CONTROL

Asynchronous data transfer are handled using the following control signals: address strobe, read/write, upper and lower data strobes, and data transfer acknowledge. These signals are explained in the following paragraphs.

Address Strobe ($\overline{AS}$)

This signal indicates that there is a valid address on the address bus.

Read/Write (R/$\overline{W}$)

This signal defines the data bus transfer as a read or write cycle. The R/$\overline{W}$ signal also works in conjunction with the upper and lower data strobes as explained in the following paragraph.

Upper and Lower Data Strobes ($\overline{UDS}$, $\overline{LDS}$)

These signals control the data on the data bus, as shown in Table 13. When the R/$\overline{W}$ line is high, the processor will read from the data bus as indicated. When the R/$\overline{W}$ line is low, the processor will write to the data bus as shown.

Table 13 Data Strobe Control of Data Bus

$\overline{UDS}$	$\overline{LDS}$	R/$\overline{W}$	$D_8 \sim D_{15}$	$D_0 \sim D_7$
High	High	–	No valid data	No valid data
Low	Low	High	Valid data bits 8 ~ 15	Valid data bits 0 ~ 7
High	Low	High	No valid data	Valid data bits 0 ~ 7
Low	High	High	Valid data bits 8 ~ 15	No valid data
Low	Low	Low	Valid data bits 8 ~ 15	Valid data bits 0 ~ 7
High	Low	Low	Valid data bits 0 ~ 7*	Valid data bits 0 ~ 7
Low	High	Low	Valid data bits 8 ~ 15	Valid data bits 8 ~ 15*

* These conditions are a result of current implementation and may not appear on future devices

Data Transfer Acknowledge ($\overline{DTACK}$)

This input indicates that the data transfer is completed. When the processor recognizes $\overline{DTACK}$ during a read cycle, data is latched and the bus cycle terminated. When $\overline{DTACK}$ is recognized during a write cycle, the bus cycle is terminated. (Refer to ASYNCHRONOUS VERSUS SYNCHRONOUS OPERATION)

BUS ARBITRATION CONTROL

These three signals form a bus arbitration circuit to determine which device will be the bus master device.

Bus Request ($\overline{BR}$)

This input is wire ORed with all other devices that could be bus masters. This input indicates to the processor that some other device desires to become the bus master.

Bus Grant (BG)

This output indicates to all other potential bus master devices that the processor will release bus control at the end of the current bus cycle.

Bus Grand Acknowledge ($\overline{BGACK}$)

This input indicates that some other device has become the bus master. This signal cannot be asserted until the following four conditions are met

(1) A Bus Grant has been received
(2) Address Strobe is inactive which indicates that the microprocessor is not using the bus
(3) Data Transfer Acknowledge is inactive which indicates

that neither memory nor peripherals are using the bus
(4) Bus Grant Acknowledge is inactive which indicates that no other device is still claiming bus mastership.

INTERRUPT CONTROL ($\overline{IPL_0}$, $\overline{IPL_1}$, $\overline{IPL_2}$)

These input pins indicate the encoded priority level of the device requesting an interrupt. Level seven is the highest priority while level zero indicates that no interrupts are requested. Level seven can not be masked. The least significant bit is given in $\overline{IPL_0}$ and the most significant bit is contained in $\overline{IPL_2}$. These lines must remain stable until the processor signals interrupt acknowledge ($FC_0 \sim FC_2$ are all high) to insure that the interrupt is recognized.

SYSTEM CONTROL

The system control inputs are used to either reset or halt the processor and to indicate to the processor that bus errors have occurred. The three system control inputs are explained in the following paragraphs.

Bus Error ($\overline{BERR}$)

This input informs the processor that there is a problem with the cycle currently being executed. Problems may be a result of:

(1) Nonresponding devices
(2) Interrupt vector number acquisition failure
(3) Illegal access request as determined by a memory management unit
(4) Other application dependent errors.

The bus error signal interacts with the halt signal to determine if exception processing should be performed or if the current bus cycle should be retried.

Refer to **BUS ERROR AND HALT OPERATION** paragraph for additional information about the interaction of the bus error and halt signals.

Reset ($\overline{RES}$)

This bidirectional signal line acts to reset (initiate a system initialization sequence) the processor in response to an external reset signal. An internally generated reset (result of a RESET instruction) causes all external devices to be reset and the internal state of the processor is not affected. A total system reset (processor and external devices) is the result of external HALT and RESET signals applied at the same time. Refer to **RESET OPERATION** paragraph for additional information about reset operation.

Halt ($\overline{HALT}$)

When this bidirectional line is driven by an external device, it will cause the processor to stop at the completion of the current bus cycle. When the processor has been halted using this input, all control signals are inactive and all three-state lines are put in their high-impedance state. Refer to **BUS ERROR AND HALT OPERATION** paragraph for additional information about the interaction between the halt and bus error signals.

When the processor has stopped executing instructions, such as in a double bus fault condition, the halt line is driven by the processor to indicate to external devices that the processor has stopped.

HMCS6800 PERIPHERAL CONTROL

These control signals are used to allow the interfacing of synchronous HMCS6800 peripheral devices with the asynchronous HD68000. These signals are explained in the following paragraphs.

Enable (E)

This signal is the standard enable signal common to all HMCS6800 type peripheral devices. The period for this output is ten HD68000 clock periods (six clocks low; four clocks high). Enable is generated by an internal ring counter which may come up in any state (i.e., at power on, it is impossible to guarantee phase relationship of E to CLK). E is a free-running clock and runs regardless of the state of the bus on the MPU.

Valid Peripheral Address ($\overline{VPA}$)

This input indicates that the device or region addressed is a HMCS6800 family device and that data transfer should be synchronized with the enable (E) signal. This input also indicates that the processor should use automatic vectoring for an interrupt. Refer to **INTERFACE WITH HMCS6800 PERIPHERALS.**

Valid Memory Address ($\overline{VMA}$)

This output is used to indicate to HMCS6800 peripheral devices that there is a valid address on the address bus and the processor is synchronized to enable. This signal only responds to a valid peripheral address ($\overline{VPA}$) input which indicates that the peripheral is a HMCS6800 family device.

PROCESSOR STATUS (FC_0, FC_1, FC_2)

These function code outputs indicate the state (user or supervisor) and the cycle type currently being executed, as shown in Table14. The information indicated by the function code outputs is valid whenever address strobe ($\overline{AS}$) is active.

Table 14 Function Code Outputs

FC_2	FC_1	FC_0	Cycle Type
Low	Low	Low	(Undefined, Reserved)
Low	Low	High	User Data
Low	High	Low	User Program
Low	High	High	(Undefined, Reserved)
High	Low	Low	(Undefined, Reserved)
High	Low	High	Superviser Data
High	High	Low	Supervisor Program
High	High	High	Interrupt Acknowledge

CLOCK (CLK)

The clock input is a TTL-compatible signal that is internally buffered for development of the internal clocks needed by the processor. The clock input should not be gated off at any time, and the clock signal must conform to minimum and maximum pulse width time.

SIGNAL SUMMARY

Table 15 is a summary of all the signals discussed in the previous paragraphs.

• BUS OPERATION

The following paragraphs explain control signal and bus operation during data transfer operations, bus arbitration, bus error and halt conditions, and reset operation.

Table 15 Signal Summary

Signal Name	Mnemonic	Input/Output	Active State	Three State	
				On $\overline{BGACK}$	On $\overline{HALT}$
Address Bus	$A_1 \sim A_{23}$	output	high	yes	yes
Data Bus	$D_0 \sim D_{15}$	input/output	high	yes	yes
Address Strobe	$\overline{AS}$	output	low	yes	no
Read/Write	$R/\overline{W}$	output	read-high write-low	yes	no
Upper and Lower Data Strobes	$\overline{UDS}$, $\overline{LDS}$	output	low	yes	no
Data Transfer Acknowledge	$\overline{DTACK}$	input	low	no	no
Bus Request	$\overline{BR}$	input	low	no	no
Bus Grant	$\overline{BG}$	output	low	no	no
Bus Grant Acknowledge	$\overline{BGACK}$	input	low	no	no
Interrupt Priority Level	$\overline{IPL_0}$, $\overline{IPL_1}$, $\overline{IPL_2}$	input	low	no	no
Bus Error	$\overline{BERR}$	input	low	no	no
Reset	$\overline{RES}$	input/output	low	no*	no*
Halt	$\overline{HALT}$	input/output	low	no*	no*
Enable	E	output	high	no	no
Valid Memory Address	$\overline{VMA}$	output	low	yes	no
Valid Peripheral Address	$\overline{VPA}$	input	low	no	no
Function Code Output	FC_0, FC_1, FC_2	output	high	yes	no
Clock	CLK	input	high	no	no
Power Input	V_{CC}	input	–	–	–
Ground	V_{SS}	input	–	–	–

* Open drain

DATA TRANSFER OPERATIONS

Transfer of data between devices involve the following leads:

(1) Address Bus A_1 through A_{23}
(2) Data Bus D_0 through D_{15}
(3) Control Signals

The address and data buses are separate parallel buses used to transfer data using an asynchronous bus structure. In all cycles, the bus master assumes responsibility for deskewing all signals it issues at both the start and end of a cycle. In addition, the bus master is responsible for deskewing the acknowledge and data signals from the slave device

The following paragraphs explain the read, write, and read-modify-write cycles. The indivisible read-modify-write cycle is the method used by the HD68000 for interlocked multiprocessor communications.

(NOTE) The terms **assertion** and **negation** will be used extensively. This is done to avoid confusion when dealing with a mixture of "active-low" and "active-high" signals. The term assert or assertion is used to indicate that a signal is active or true independent of whether that voltage is low or high. The term negate or negation is used to indicate that a signal is inactive or false.

Read Cycle

During a read cycle, the processor receives data from memory or a peripheral device. The processor reads bytes of data in all cases. If the instruction specifies a word (or double word) operation, the processor reads both upper and lower bytes simultaneously by asserting both upper and lower data strobes. When the instruction specifies byte operation, the processor uses an internal Ao bit to determine which byte to read and then issues the data strobe required for that byte. For bytes operations, when the Ao bit equals zero, the upper data strobe is issued. When the Ao bit equals one, the lower data strobe is issued. When the data is received, the processor correctly positions it internally.

A word read cycle flow chart is given in Figure 13. A byte read cycle flow chart is given in Figure 14. Read cycle timing is given in Figure 15. Figure 16 details word and byte read cycle operations. Refer to these illustrations during the following detailed.

At state zero (S0) in the read cycle, the address bus (A_1 through A_{23}) is in the high impedance state. A function code is asserted on the function code output line (FC_0 through FC_2). The read/write (R/W) signal is switched high to indicate a read cycle. One half clock cycle later, at state 1, the address bus is released from the high impedance state. The function code outputs indicate which address space that this cycle will operate on.

In state 2, the address strobe ($\overline{AS}$) is asserted to indicate that there is a valid address on the address bus and the upper and lower data strobe ($\overline{UDS}$, $\overline{LDS}$) is asserted as required. The memory or peripheral device uses the address bus and the address strobe to determine if it has been selected. The selected device uses the read/write signal and the data strobe to place its information on the data bus. Concurrent with placing data on the data bus, the selected device asserts data transfer acknowledge ($\overline{DTACK}$).

Data transfer acknowledge must be present at the processor at the start of state 5 or the processor will substitute wait states for states 5 and 6. State 5 starts the synchronization of the returning data transfer acknowledge. At the end of state 6 (beginning of state 7) incoming data is latched into an internal data bus holding register.

During state 7, address strobe and the upper and/or lower data strobes are negated. The address bus is held valid through state 7 to allow for static memory operation and signal skew.

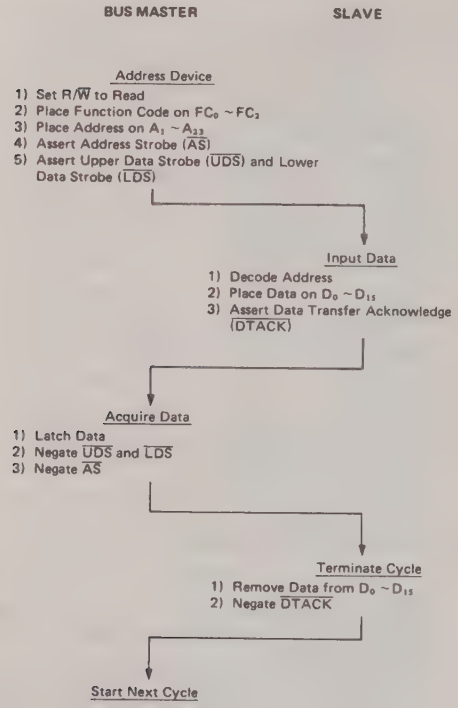

BUS MASTER
SLAVE
Address Device
1) Set R/W to Read
2) Place Function Code on FC0 ~ FC2
3) Place Address on A1 ~ A23
4) Assert Address Strobe (AS)
5) Assert Upper Data Strobe (UDS) and Lower Data Strobe (LDS)
Input Data
1) Decode Address
2) Place Data on D0 ~ D15
3) Assert Data Transfer Acknowledge (DTACK)
Acquire Data
1) Latch Data
2) Negate UDS and LDS
3) Negate AS
Terminate Cycle
1) Remove Data from D0 ~ D15
2) Negate DTACK
Start Next Cycle

Figure 13 Word Read Cycle Flow Chart

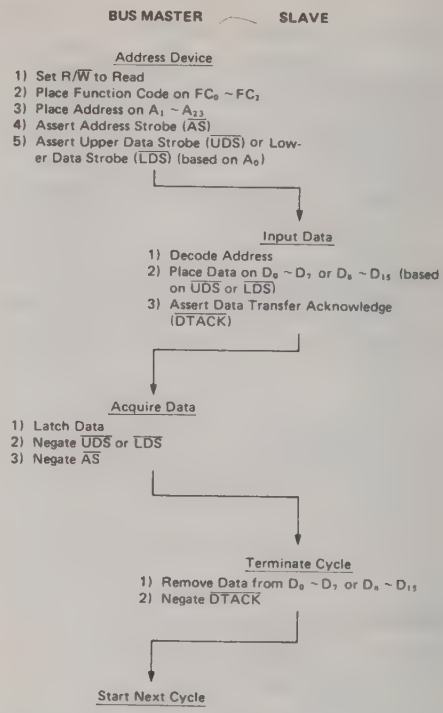

BUS MASTER
SLAVE
Address Device
1) Set R/W to Read
2) Place Function Code on FC0 ~ FC2
3) Place Address on A1 ~ A23
4) Assert Address Strobe (AS)
5) Assert Upper Data Strobe (UDS) or Lower Data Strobe (LDS) (based on A0)
Input Data
1) Decode Address
2) Place Data on D0 ~ D7 or D8 ~ D15 (based on UDS or LDS)
3) Assert Data Transfer Acknowledge (DTACK)
Acquire Data
1) Latch Data
2) Negate UDS or LDS
3) Negate AS
Terminate Cycle
1) Remove Data from D0 ~ D7 or D8 ~ D15
2) Negate DTACK
Start Next Cycle

Figure 14 Byte Read Cycle Flow Chart

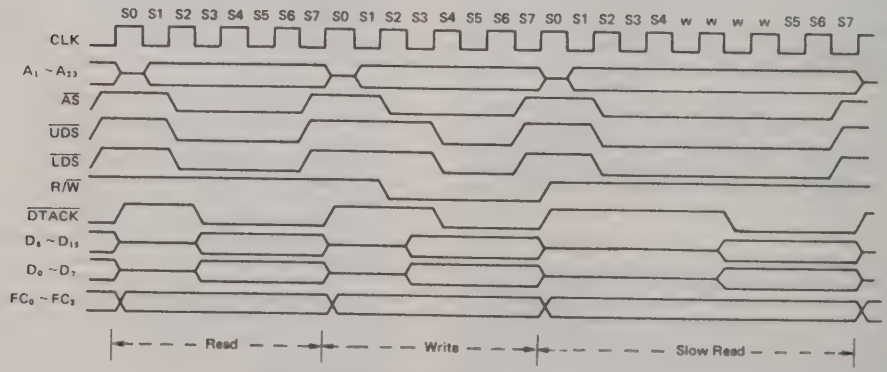


Figure 15 Read and Write Cycle Timing Diagram

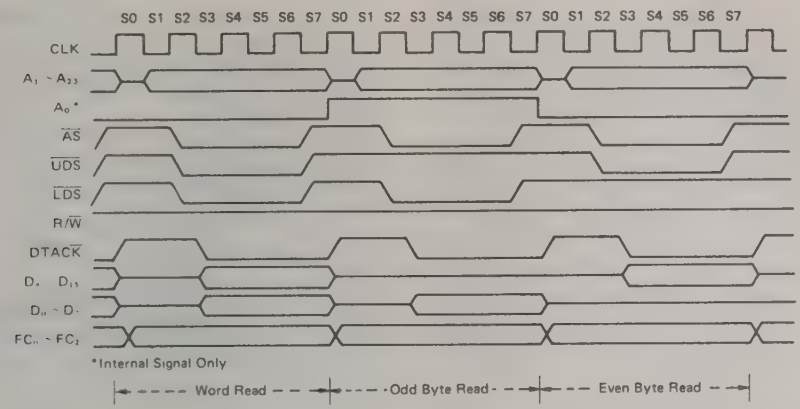


Figure 16 Word and Byte Read Cycle Timing Diagram

The read/write signal and the function code outputs also remain valid through state 7 to ensure a correct transfer operation. The slave device keeps its data asserted until it detects the negation of either the address strobe or the upper and/or lower data strobe. The slave device must remove its data and data transfer acknowledge within one clock period of recognizing the negation of the address or data strobes. Note that the data bus might not become free and data transfer acknowledge might not be removed until state 0 or 1.

When address strobe is negated, the slave device is released. Note that a slave device must remain selected as long as address strobe is asserted to ensure the correct functioning of the read-modify-write cycle.

Write Cycle

During a write cycle, the processor sends data to memory or a peripheral device. The processor writes bytes of data in all cases. If the instruction specifies a word operation, the processor writes both bytes. When the instruction specifies a byte operation, the processor uses an internal A_0 bit to determine which byte to write and then issues the data strobe required for that byte. For byte operations, when the A_0 bit equals zero, the upper data strobe is issued. When the A_0 bit equals one, the lower data strobe is issued. A word write cycle flow chart is given in Figure 17. A byte write cycle flow chart is given in Figure 18. Write cycle timing is given in Figure 15. Figure 19 details word and byte write cycle operation. Refer to these illustrations during the following detailed discussion.

At state zero (S0) in the write cycle, the address bus (A_1 through A_{23}) is in the high impedance state. A function code is asserted on the function code output line (FC_0 through FC_2).

(NOTE) The read/write (R/$\overline{W}$) signal remains high until state 2 to prevent bus conflicts with preceding read cycles. The data bus is not driven until state 3.

One half clock later, at state 1, the address bus is released from the high impedance state. The function code outputs indicate which address space that this cycle will operate on.

In state 2, the address strobe ($\overline{AS}$) is asserted to indicate that there is a valid address on the address bus. The memory or peripheral device uses the address bus and the address strobe to determine if it has been selected. During state 2, the read/write signal is switched low to indicate a write cycle. When external processor data bus buffers are required, the read/write line provides sufficient directional control. Data is not asserted during this state to allow sufficient turn around time for external data buffers (if used). Data is asserted onto the data bus during state 3.

In state 4, the data strobes are asserted as required to indicate that the data bus is stable. The selected device uses the read/write signal and the data strobes to take its information from the data bus. The selected device asserts data transfer acknowledge ($\overline{DTACK}$) when it has successfully stored the data.

Data transfer acknowledge must be present at the processor at the start of state 5 or the processor will substitute wait states for states 5 and 6. State 5 starts the synchronization of the returning data transfer acknowledge.

During state 7, address strobe and the upper and/or lower data strobes are negated. The address and data buses are held valid through state 7 to allow for static memory operation and signal skew. The read/write signal and the function code outputs also remain valid through state 7 to ensure a correct transfer operation. The slave device keeps its data transfer acknowledge asserted until it detects the negation of either the address strobe or the upper and/or lower data strobe. The slave device must remove its data transfer acknowledge within one clock period after recognizing the negation of the address or data strobes. Note that the processor releases the data bus at the end of state 7 but that data transfer acknowledge might not be removed until state 0 or 1. When address strobe is negated, the slave device is released.

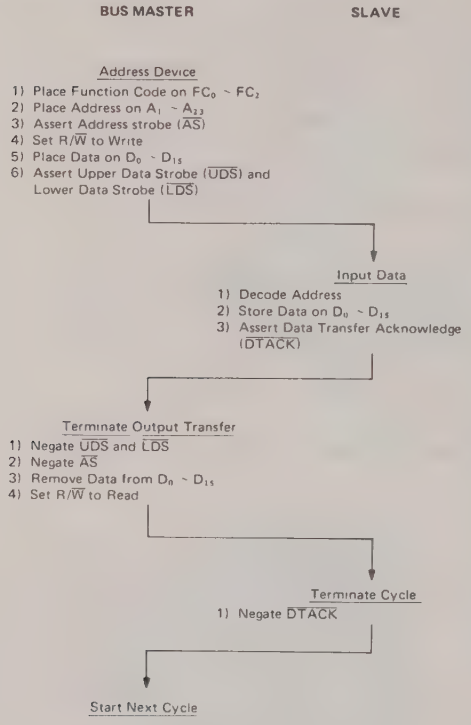

BUS MASTER
SLAVE
Address Device
1) Place Function Code on FC0 ~ FC2
2) Place Address on A1 ~ A23
3) Assert Address strobe (AS)
4) Set R/W to Write
5) Place Data on D0 ~ D15
6) Assert Upper Data Strobe (UDS) and Lower Data Strobe (LDS)
Input Data
1) Decode Address
2) Store Data on D0 ~ D15
3) Assert Data Transfer Acknowledge (DTACK)
Terminate Output Transfer
1) Negate UDS and LDS
2) Negate AS
3) Remove Data from D0 ~ D15
4) Set R/W to Read
Terminate Cycle
1) Negate DTACK
Start Next Cycle

Figure 17 Word Write Cycle Flow Chart

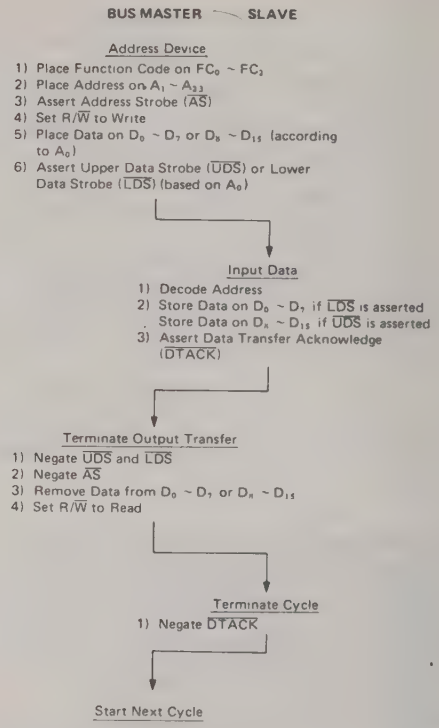

BUS MASTER
SLAVE
Address Device
1) Place Function Code on FC0 ~ FC2
2) Place Address on A1 ~ A23
3) Assert Address Strobe (AS)
4) Set R/W to Write
5) Place Data on D0 ~ D7 or Dn ~ D15 (according to A0)
6) Assert Upper Data Strobe (UDS) or Lower Data Strobe (LDS) (based on A0)
Input Data
1) Decode Address
2) Store Data on D0 ~ D7 if LDS is asserted
Store Data on Dn ~ D15 if UDS is asserted
3) Assert Data Transfer Acknowledge (DTACK)
Terminate Output Transfer
1) Negate UDS and LDS
2) Negate AS
3) Remove Data from D0 ~ D7 or Dn ~ D15
4) Set R/W to Read
Terminate Cycle
1) Negate DTACK
Start Next Cycle

Figure 18 Byte Write Cycle Flow Chart

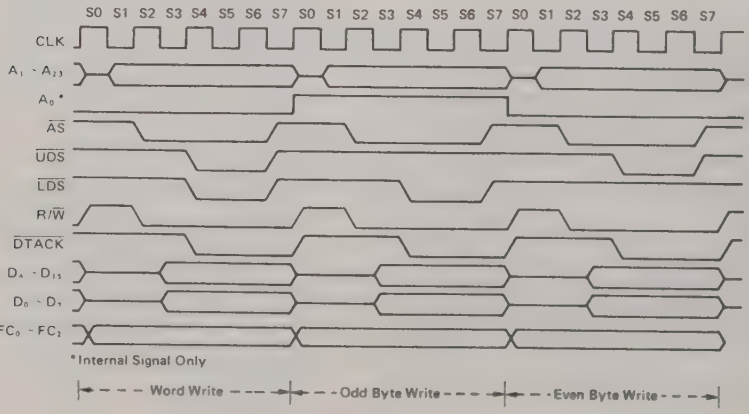


Figure 19 Word and Byte Write Cycle Timing Diagram

Read-Modify-Write Cycle

The read-modify-write cycle performs a read, modifies the data in the arithmetic-logic unit, and writes the data back to the same address. In the HD68000 this cycle is indivisible in that the address strobe is asserted throughout the entire cycle. The test and set (TAS) instruction uses this cycle to provide meaningful communication between processors in a multiple processor environment. This instruction is the only instruction that uses the read-modify-write cycle and since the test and set instruction only operates on bytes, all read-modify-write cycles are byte operations. A read-modify-write cycle flow chart is given in Figure 20 and a timing diagram is given in Figure 21. Refer to these illustrations during the following detailed discussions.

At state zero (S0) in the read-modify-write cycle, the address bus (A_1 through A_{23}) is in the high impedance state. A function code is asserted on the function code output line (FC_0 through FC_2). The read/write (R/$\overline{W}$) signal is switched high to indicate a read cycle. One half clock cycle later, at state 1, the address bus is released from the high impedance state. The function code outputs indicate which address space that this cycle will operate on.

In state 2, the address strobe (AS) is asserted to indicate that there is a valid address on the address bus and the upper or lower data strobe ($\overline{UDS}$, $\overline{LDS}$) is asserted as required. The memory or peripheral device uses the address bus and the address strobe to determine if it has been selected. The selected device uses the read/write signal and the data strobe to place its information on the data bus. Concurrent with placing data on the data bus, the selected device asserts data transfer acknowledge ($\overline{DTACK}$).

Data transfer acknowledge must be present at the processor at the start of state 5 or the processor will substitute wait states for states 5 and 6. State 5 starts the synchronization of the returning data transfer acknowledge. At the end of state 6 (beginning of state 7) incoming data is latched into an internal data bus holding register.

During state 7, the upper or lower data strobe is negated. The address bus, address strobe, read/write signal, and function code outputs remain as they were in preparation for the write portion of the cycle. The slave device keeps its data asserted until it detects the negation of the upper or lower data strobe. The slave device must remove its data and data transfer acknowledge within one clock period of recognizing the negation of the data strobes. Internal modification of data may occur from state 8 to state 11.

(NOTE) The read/write signal remains high until state 14 to prevent bus conflicts with the preceding read portion of the cycle and the data bus is not asserted by the processor until state 15.

In state 14, the read/write signal is switched low to indicate a write cycle. When external processor data bus buffers are required, the read/write line provides sufficient directional control. Data is not asserted during this state to allow sufficient turn around time for external data buffers (if used). Data is asserted onto the data bus during state 15.

In state 16, the data strobe is asserted as required to indicate that the data bus is stable. The selected device uses the read/write signal and the data strobe to take its information from the data bus. The selected device asserts data transfer acknowledge ($\overline{DTACK}$) when it has successfully stored its data.

Data transfer acknowledge must be present at the processor at the start of state 17 or the processor will substitute wait states for states 17 and 18. State 17 starts the synchronization of the returning data transfer acknowledge for the write portion of the cycle. The bus interface circuitry issues requests for subsequent internal cycles during state 18.

During state 19, address strobe and the upper or lower data strobe is negated. The address and data buses are held valid through state 19 to allow for static memory operation and signal skew. The read/write signal and the function code outputs also remain valid through state 19 to ensure a correct transfer operation. The slave device keeps its data transfer acknowledge asserted until it detects the negation of either the address strobe or the upper or lower data strobe. The slave device must remove its data transfer acknowledge within once clock period after recognizing the negation of the address or data strobes. Note that the processor releases the data bus at the end of state 19 but that data transfer acknowledge might not be removed until state 0 or 1. When address strobe is negated the slave device is released.

BUS ARBITRATION

Bus arbitration is a technique used by master-type devices to request, be granted, and acknowledge bus mastership. In its simplest form, it consists of:

(1) Asserting a bus mastership request.
(2) Receiving a grant that the bus is available at the end of the current cycle.
(3) Acknowledging that mastership has been assumed.

Figure 22 is a flow chart showing the detail involved in a request from a single device. Figure 23 is a timing diagram for the same operations. This technique allows processing of bus requests during data transfer cycles.

The timing diagram shows that the bus request is negated at the time that an acknowledge is asserted. This type of operation would be true for a system consisting of the processor and one device capable of bus mastership. In systems having a number of devices capable of bus mastership, the bus request line from each device is wire ORed to the processor. In this system, it is easy to see that there could be more than one bus request being made. The timing diagram shows that the bus grant signal is negated a few clock cycles after the transition of the acknowledge ($\overline{BGACK}$) signal.

However, if the bus requests are still pending, the processor will assert another bus grant within a few clock cycles after it was negated. This additional assertion of bus grant allows external arbitration circuitry to select the next bus master before the current bus master has completed its requirements. The following paragraphs provide additional information about the three steps in the arbitration process.

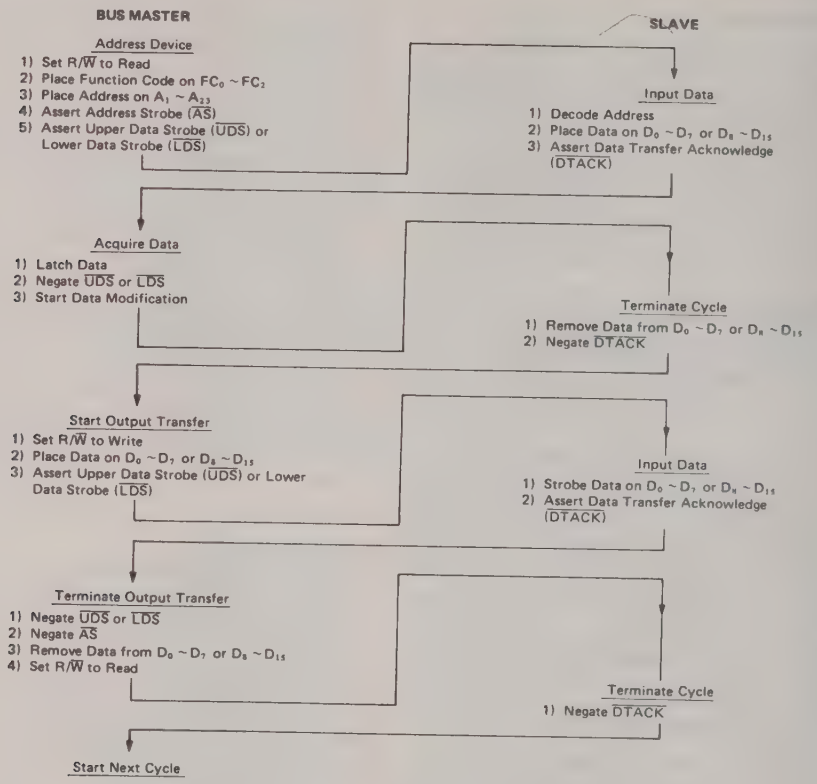

BUS MASTER
SLAVE
Address Device
1) Set R/W to Read
2) Place Function Code on FC0 ~ FC2
3) Place Address on A1 ~ A23
4) Assert Address Strobe (AS)
5) Assert Upper Data Strobe (UDS) or Lower Data Strobe (LDS)
Input Data
1) Decode Address
2) Place Data on D0 ~ D7 or D8 ~ D15
3) Assert Data Transfer Acknowledge (DTACK)
Acquire Data
1) Latch Data
2) Negate UDS or LDS
3) Start Data Modification
Terminate Cycle
1) Remove Data from D0 ~ D7 or D8 ~ D15
2) Negate DTACK
Start Output Transfer
1) Set R/W to Write
2) Place Data on D0 ~ D7 or D8 ~ D15
3) Assert Upper Data Strobe (UDS) or Lower Data Strobe (LDS)
Input Data
1) Strobe Data on D0 ~ D7 or D8 ~ D15
2) Assert Data Transfer Acknowledge (DTACK)
Terminate Output Transfer
1) Negate UDS or LDS
2) Negate AS
3) Remove Data from D0 ~ D7 or D8 ~ D15
4) Set R/W to Read
Terminate Cycle
1) Negate DTACK
Start Next Cycle

Figure 20 Read-Modify-Write Cycle Flow Chart

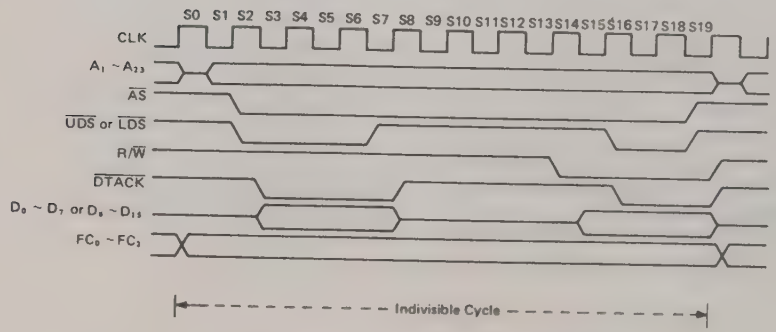

S0 S1 S2 S3 S4 S5 S6 S7 S8 S9 S10 S11 S12 S13 S14 S15 S16 S17 S18 S19
CLK
A1 ~ A23
AS
UDS or LDS
R/W
DTACK
D0 ~ D7 or D8 ~ D15
FC0 ~ FC2
Indivisible Cycle

Figure 21 Read-Modify-Write Cycle Timing Diagram

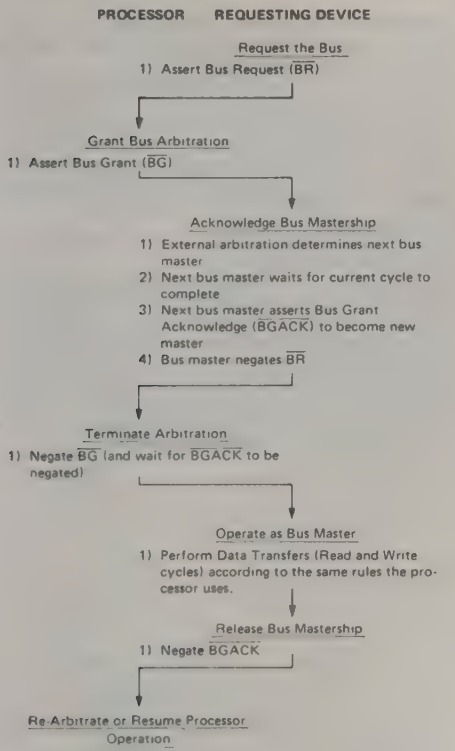

PROCESSOR
REQUESTING DEVICE
Request the Bus
1) Assert Bus Request (BR)
Grant Bus Arbitration
1) Assert Bus Grant (BG)
Acknowledge Bus Mastership
1) External arbitration determines next bus master
2) Next bus master waits for current cycle to complete
3) Next bus master asserts Bus Grant Acknowledge (BGACK) to become new master
4) Bus master negates BR
Terminate Arbitration
1) Negate BG (and wait for BGACK to be negated)
Operate as Bus Master
1) Perform Data Transfers (Read and Write cycles) according to the same rules the processor uses.
Release Bus Mastership
1) Negate BGACK
Re-Arbitrate or Resume Processor Operation

Figure 22 Bus Arbitration Cycle Flow Chart

Requesting the Bus

External devices capable of becoming bus masters request the bus by asserting the bus request ($\overline{BR}$) signal. This is a wire ORed signal (although it need not be constructed from open collector devices) that indicates to the processor that some external device requires control of the external bus. The processor is effectively at a lower bus priority level that the external device and will relinquish the bus after it has completed the last bus cycle it has started.

When no acknowledge is received before the bus request signal goes inactive, the processor will continue processing when it detects that the bus request is inactive. This allows ordinary processing to continue if the arbitration circuitry responded to noise inadvertently.

Receiving the Bus Grant

The processor asserts bus grant ($\overline{BG}$) as soon as possible. Normally this is immediately after internal synchronization. The only exception to this occurs when the processor has made an internal decision to execute the next bus cycle but has not progressed far enough into the cycle to have asserted the address strobe ($\overline{AS}$) signal. In this case, bus grant will not be asserted until one clock after address strobe is asserted to indicate to external devices that a bus cycle is being executed.

The bus grant signal may be routed through a daisy-chained network or through a specific priority-encoded network. The processor is not affected by the external method of arbitration as long as the protocol is obeyed.

Acknowledgement of Mastership

Upon receiving a bus grant, the requesting device waits until address strobe, data transfer acknowledge, and bus grant acknowledge are negated before issuing its own $\overline{BGACK}$. The negation of the address strobe indicates that the previous master has completed its cycle, the negation of bus grant acknowledge indicates that the previous master has released the bus. (While address strobe is asserted no device is allowed to "break into" a cycle.) The negation of data transfer acknowledge indicates the previous slave has terminated its connection to the previous master. Note that in some applications data

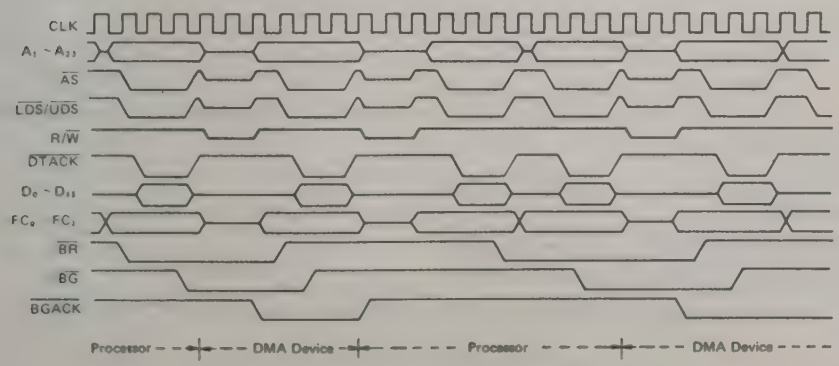


Figure 23 Bus Arbitration Cycle Timing Diagram

transfer acknowledge might not enter into this function. General purpose devices would then be connected such that they were only dependent on address strobe. When bus grant acknowledge is issued the device is bus master until it negates bus grant acknowledge. Bus grant acknowledge should not be negated until after the bus cycle(s) is (are) completed. Bus mastership is terminated at the negation of bus grant acknowledge.

The bus request from the granted device should be dropped after bus grant acknowledge is asserted. If a bus request is still pending, another bus grant will be asserted within a few clocks of the negation of bus grant. Refer to Bus Arbitration Control section. Note that the processor does not perform any external bus cycles before it re-asserts bus grant.

BUS ARBITRATION CONTROL

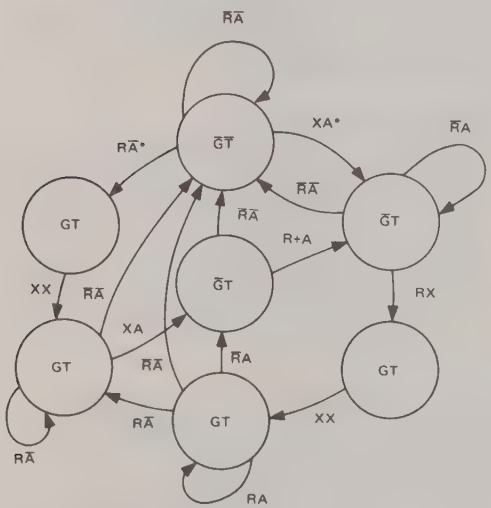
The bus arbitration control unit in the HD68000 is implemented with a finite state machine. A state diagram of this machine is shown in Figure 24. All asynchronous signals to the HD68000 are synchronized before being used internally. This synchronization is accomplished in a maximum of one cycle of the system clock, assuming that the asynchronous input setup time (#47) has been met (see Figure 25). The input signal is sampled on the falling edge of the clock and is valid internally after the next falling edge.

As shown in Figure 24, input signals labeled R and A are internally synchronized on the bus request and bus grant acknowledge pins respectively. The bus grant output is lebeled G and the internal three-state control signal T. If T is true, the address, data, function code line, and control buses are placed in a high-impedance state when $\overline{AS}$ is negated. All signals are shown in positive logic (active high) regardless of their true active voltage level.

State changes (valid outputs) occur on the next rising edge after the internal signal is valid.

A timing diagram of the bus arbitration sequence during a processor bus cycle is shown in Figure 26. The bus arbitration sequence while the bus is inactive (i.e., executing internal operations such as a multiply instruction) is shown in Figure 27.

If a bus request is made at a time when the MPU has already begun a bus cycle but $\overline{AS}$ has not been asserted (bus state S0), $\overline{BG}$ will not be asserted on the next rising edge. Instead, $\overline{BG}$ will be delayed until the second rising edge following it's internal assertion. This sequence is shown in Figure 28.



R = Bus Request Internal
A = Bus Grant Acknowledge Internal
G = Bus Grant
T = Three-State Control to Bus Control Logic**
X = Don't Care

* State machine will not change state if bus is in S0. Refer to BUS ARBITRATION CONTROL for additional information.
** The address bus will be placed in the high impedance state if T is asserted and $\overline{AS}$ is negated.

Figure 24 State Diagram of HD68000 Bus Arbitration Unit

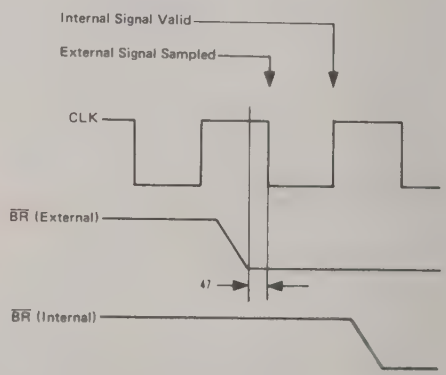


Figure 25 Timing Relationship of External Asynchronous Inputs to Internal Signals

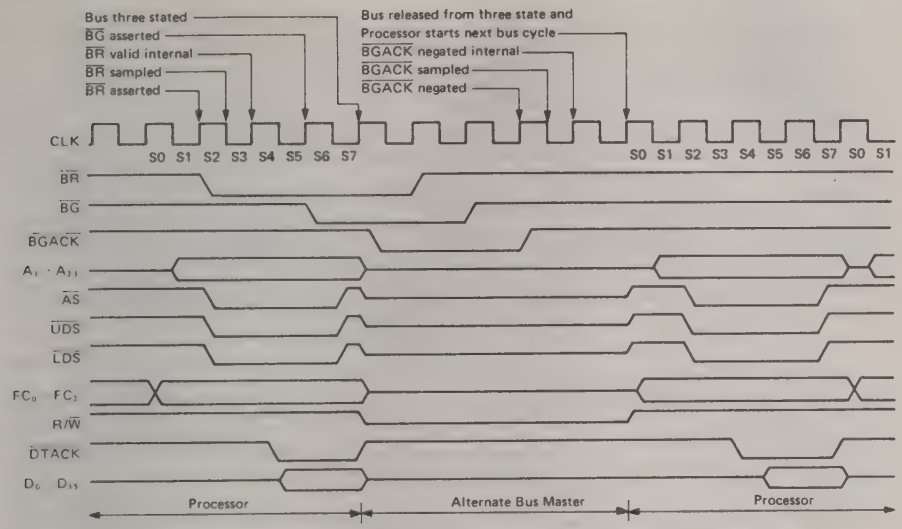


Figure 26 Bus Arbitration During Processor Bus Cycle

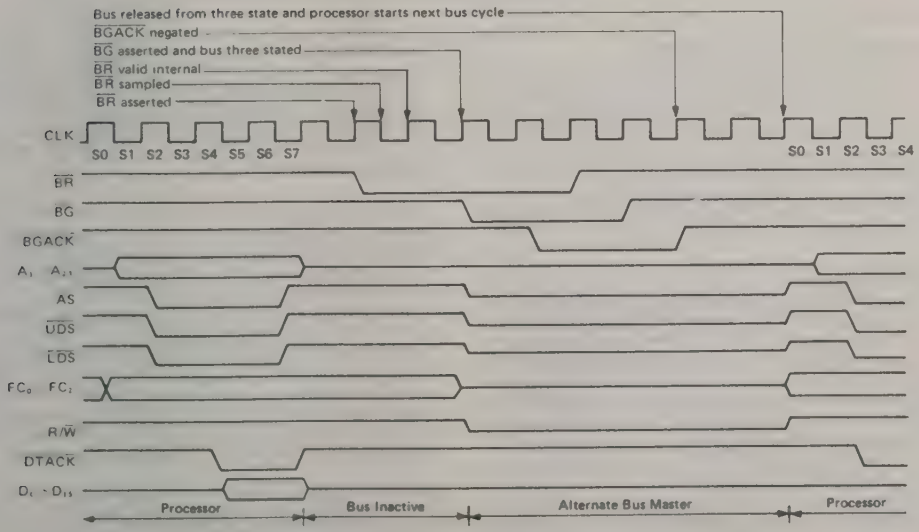


Figure 27 Bus Arbitration with Bus Inactive

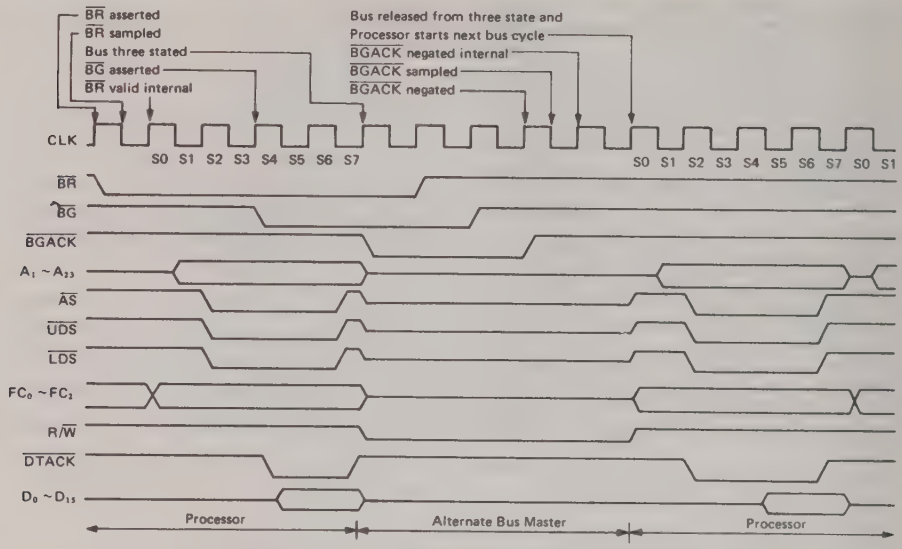


Figure 28 Bus Arbitration During Processor Bus Cycle Special Case

BUS ERROR AND HALT OPERATION

In a bus architecture that requires a handshake from an external device, the possibility exists that the handshake might not occur. Since different systems will require a different maximum response time, a bus error input is provided. External circuitry must be used to determine the duration between address strobe and data transfer acknowledge before issuing a bus error signal. When a bus error signal is received, the processor has two options initiate a bus error exception sequence or try running the bus cycle again.

Exception Sequence

When the bus error signal is asserted, the current bus cycle is terminated. If BERR is asserted before the falling edge of S2, AS will be negated in S7 in either a read or write cycle. As long as BERR remains asserted, the data and address buses will be in the high-impedance state. When BERR is negated, the processor will begin stacking for exception processing. Figure 29 is a timing diagram for the exception sequence. The sequence is composed of the following elements.

(1) Stacking the program counter and status register
(2) Stacking the error information
(3) Reading the bus error vector table entry
(4) Executing the bus error handler routine

The stacking of the program counter and the status register is the same as if an interrupt had occurred. Several additional items are stacked when a bus error occurs. These items are used to determine the nature of the error and correct it, if possible. The bus error vector is vector number two located at address $000008. The processor loads the new program counter from this location. A software bus error handler routine is then executed by the processor. Refer to **EXCEPTION PROCESSING** for additional information.

Re-Running the Bus Cycle

When, during a bus cycle, the processor receives a bus error signal and the halt pin is being driven by an external device, the processor enters the re-run sequence. Figure 30 is a timing diagram for re-running the bus cycle.

The processor terminates the bus cycle, then puts the address and data output lines in the high-impedance state. The processor remains "halted," and will not run another bus cycle until the halt signal is removed by external logic. Then the processor will re-run the previous bus cycle using the same address, the same function codes, the same data (for a write operation), and the same controls. The bus error signal should be removed at least one clock cycle before the halt signal is removed.

(NOTE) The processor will not re-run a read-modify-write cycle. This restriction is made to guarantee that the entire cycle runs correctly and that the write operation of a Test-and-Set operation is performed without ever releasing AS. If BERR and HALT are asserted during a read-modify-write bus cycle, a bus error operation results.

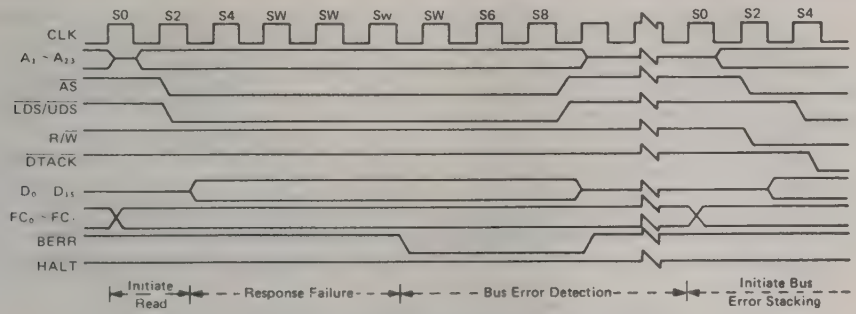


Figure 29 Bus Error Timing Diagram

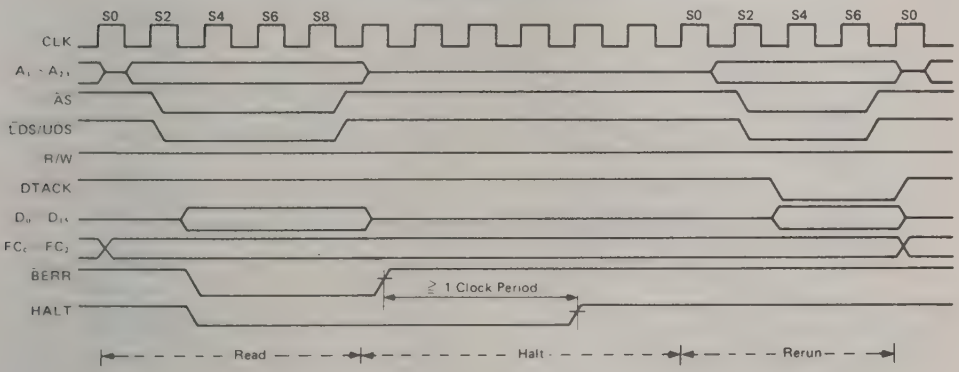


Figure 30 Re-Run Bus Cycle Timing Information

Halt Operation with No Bus Error

The halt input signal to the HD68000 perform a Halt/Run/Single-Step function in a similar fashion to the HMCS6800 halt function. The halt and run modes are somewhat self explanatory in that when the halt signal is constantly active the processor "halts" (does nothing) and when the halt signal is constantly inactive the processor "runs" (does something).

The single-step mode is derived from correctly timed transitions on the halt signal input. It forces the processor to execute a single bus cycle by entering the "run" mode until the processor starts a bus cycle then changing to the "halt" mode. Thus, the single-step mode allows the user to proceed through (and therefore debug) processor operations one bus cycle at a time.

Figure 31 details the timing required for correct single-step operations and Figure 32 shows a simple circuit for providing the single-step function. Some care must be exercised to avoid harmful interactions between the bus error signal and the halt pin when using the single cycle mode as a debugging tool. This is also true of interactions between the halt and reset lines since these can reset the machine.

When the processor completes a bus cycle after recognizing that the halt signal is active, most three-state signals are put in the high-impedance state. These include:

(1) Address lines
(2) Data lines

This is required for correct performance of the re-run bus cycle operation.

While the processor is honoring the halt request, bus arbitration performs as usual. That is, halting has no effect on bus arbitration. It is the bus arbitration function that removes the control signals from the bus.

The halt function and the hardware trace capability allow the hardware debugger to trace single bus cycles or single instructions at a time. These processor capabilities, along with a software debugging package, give total debugging flexibility.

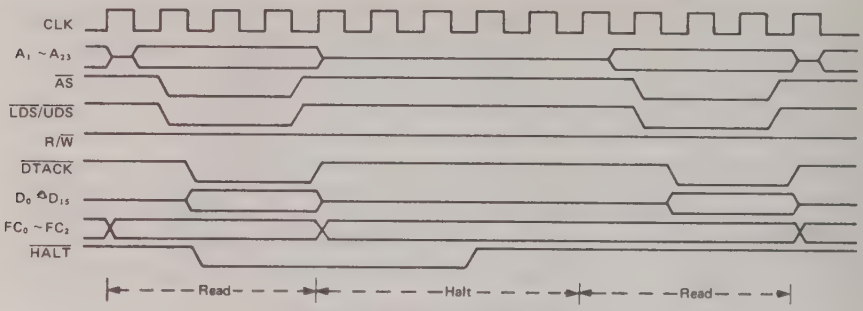


Figure 31 Halt Signal Timing Characteristics

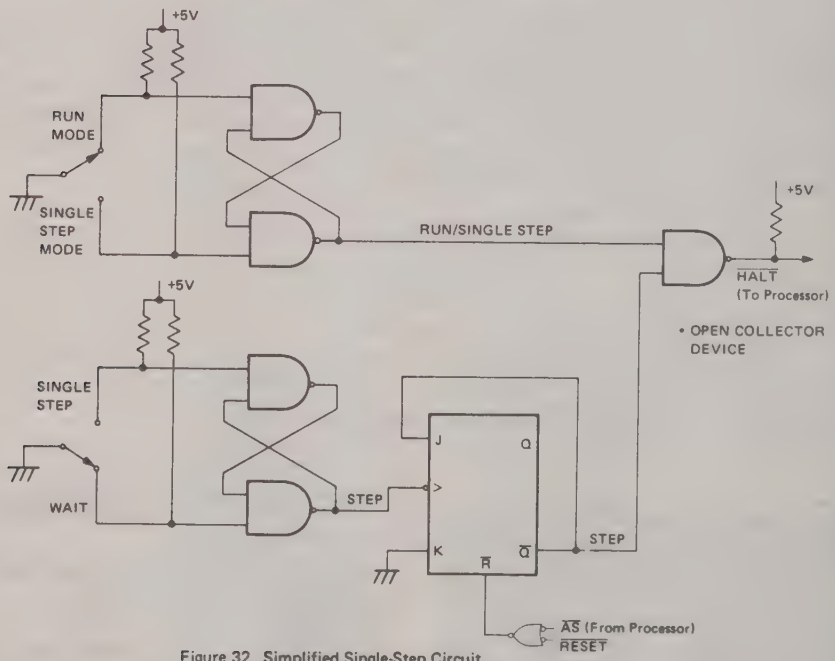


Figure 32 Simplified Single-Step Circuit

Double Bus Faults

When a bus error exception occurs, the processor will attempt to stack several words containing information about the state of the machine. If a bus error exception occurs during the stacking operation, there have been two bus errors in a row. This is commonly referred to as a double bus fault. When a double bus fault occurs, the processor will halt. Once a bus error exception has occurred, any bus error exception occurring before the execution of the next instruction constitutes a double bus fault.

Note that a bus cycle which is re-run does not constitute a bus error exception, and does not contribute to a double bus fault. Note also that this means that as long as the external hardware requests it, the processor will continue to re-run the same bus cycle.

The bus error pin also has an effect on processor operation after the processor receives an external reset input. The processor reads the vector table after a reset to determine the address to start program execution. If a bus error occurs while reading the vector table (or at any time before the first instruction is executed), the processor reacts as if a double bus fault has occurred and it halts. Only an external reset will start a halted processor.

RESET OPERATION

The reset signal is a bidirectional signal that allows either the processor or an external signal to reset the system. Figure 33 is a timing diagram for the reset operations. Both the halt and reset lines must be asserted to ensure total reset of the processor.

When the reset and halt lines are driven by an external device, it is recognized as an entire system reset, including the processor. The processor responds by reading the reset vector table entry (vector unumber zero, address $000000) and loads it into the supervisor stack pointer (SSP). Vector table entry number one at address $000004 is read next and loaded into the program counter. The processor initializes the status register to an interrupt level of seven. No other registers are affected by the reset sequence.

When a RESET instruction is executed, the processor drives the reset pin for 124 clock periods. In this case, the processor is trying to reset the rest of the system. Therefore, there is no effect on the internal state of the processor. All of the processor's internal registers and the status register are unaffected by the execution of a RESET instruction. All external devices connected to the reset line should be reset at the completion of the RESET instruction.

Asserting the Reset and Halt pins for 10 clock cycles will cause a processor reset, except when V_{CC} is initially applied to the processor. In this case, an external reset must be applied for 100 milliseconds.

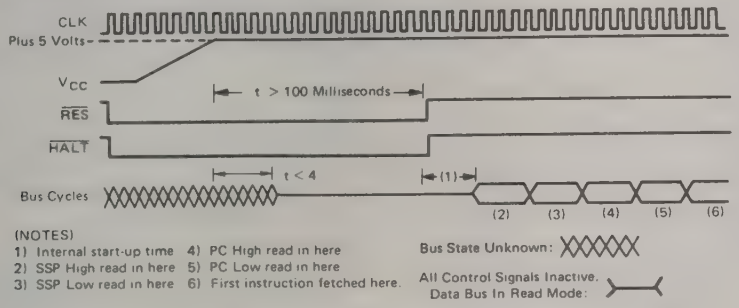


Figure 33 Reset Operation Timing Diagram

THE RELATIONSHIP OF DTACK, BERR, AND HALT

In order to properly control termination of a bus cycle for a re-run or a bus error condition, DTACK, BERR, and HALT should be asserted and negated on the rising edge of the HD68000 clock. This will assure that when two signals are asserted simultaneously, the required setup time (#47) for both of them will be met during the same bus state.

This, or some equivalent precaution, should be designed external to the HD68000. Parameter #48 is intended to ensure this operation in a totally asynchronous system, and may be ignored if the above conditions are met.

The preferred bus cycle terminations may be summarized as follows (case numbers refer to Table 16):

Normal Termination: DTACK occurs first (case 1).

Halt Termination: HALT is asserted at the same time or before DTACK and BERR remains negated (cases 2 and 3).

Bus Error Termination: BERR is asserted in lieu of, at the same time, or before DTACK (case 4); BERR is negated at the same time or after DTACK.

Re-Run Termination: HALT and BERR are asserted in lieu of, at the same time, or before DTACK (cases 6 and 7); HALT must be held at least one cycle after BERR. Case 5 indicates BERR may precede HALT which allows fully asynchronous assertion.

Table 16 details the resulting bus cycle termination under various combinations of control signal sequences. The negation of these same control signals under several conditions is shown in Table 17 (DTACK is assumed to be negated normally in all cases; for best results, both DTACK and BERR should be negated when address strobe is negated.)

Example A: A system uses a watch-dog timer to terminate accesses to un-populated address space. The timer asserts DTACK and BERR simultaneously after time-out. (case 4)

Example B: A system uses error detection on RAM contents. Designer may (a) delay DTACK until data verified, and return BERR and HALT simultaneously to re-run error cycle (case 6), or if valid, return DTACK; (b) delay DTACK until data verified, and return BERR at same time as DTACK if data in error (case 4); (c) return DTACK prior to data verification, as described in previous section. If data invalid, BERR is asserted (case 1) in next cycle. Error-handling software must know how to recover error cycle.

Table 16 $\overline{DTACK}$, $\overline{BERR}$, $\overline{HALT}$ Assertion Results

Case No.	Control Signal	Asserted on Rising Edge of State		Result
		N	N + 2	
1	$\overline{DTACK}$	A	S	Normal cycle terminate and continue.
	$\overline{BERR}$	NA	X	
	$\overline{HALT}$	NA	X	
2	$\overline{DTACK}$	A	S	Normal cycle terminate and halt. Continue when $\overline{HALT}$ removed.
	$\overline{BERR}$	NA	X	
	$\overline{HALT}$	A	S	
3	$\overline{DTACK}$	NA	A	Normal cycle terminate and halt. Continue when $\overline{HALT}$ removed.
	$\overline{BERR}$	NA	NA	
	$\overline{HALT}$	A	S	
4	$\overline{DTACK}$	X	X	Terminate and take bus error trap.
	$\overline{BERR}$	A	S	
	$\overline{HALT}$	NA	NA	
5	$\overline{DTACK}$	NA	X	Terminate and re-run.
	$\overline{BERR}$	A	S	
	$\overline{HALT}$	NA	A	
6	$\overline{DTACK}$	X	X	Terminate and re-run when $\overline{HALT}$ removed.
	$\overline{BERR}$	A	S	
	$\overline{HALT}$	A	S	
7	$\overline{DTACK}$	NA	X	Terminate and re-run when $\overline{HALT}$ removed.
	$\overline{BERR}$	NA	A	
	$\overline{HALT}$	A	S	

gend:
N – The number of the current even bus state (e.g., S4, S6, etc.)
A – Signal is asserted in this bus state
NA – Signal is not asserted in this state
X – Don't care
S – Signal was asserted in previous state and remains asserted in this state

Table 17 $\overline{BERR}$ and $\overline{HALT}$ Negation Results

Conditions of Termination in Table A	Control Signal	Negated on Rising Edge of State			Results – Next Cycle
		N		N + 2	
Bus Error	$\overline{BERR}$	●	or	●	Takes bus error trap.
	$\overline{HALT}$	●	or	●	
Re-run	$\overline{BERR}$	●	or	●	Illegal sequence; usually traps to vector number 0.
	$\overline{HALT}$	●			
Re-run	$\overline{BERR}$	●			Re-runs the bus cycle.
	$\overline{HALT}$			●	
Normal	$\overline{BERR}$	●			May lengthen next cycle.
	$\overline{HALT}$	●	or	●	
Normal	$\overline{BERR}$			●	If next cycle is started it will be terminated as a bus error.
	$\overline{HALT}$	●	or	none	

ASYNCHRONOUS VERSUS SYNCHRONOUS OPERATION

Asynchronous Operation

To achieve clock frequency independence at a system level, the HD68000 can be used in an asynchronous manner. This entails using only the bus handshake lines ($\overline{AS}$, $\overline{UDS}$, $\overline{LDS}$, $\overline{DTACK}$, $\overline{BERR}$, $\overline{HALT}$, and $\overline{VPA}$) to control the data transfer. Using this method, $\overline{AS}$ signals the start of a bus cycle and the data strobes are used as a condition for valid data on a write cycle. The slave device (memory or peripheral) then responds by placing the requested data on the data bus for a read cycle or latching data on a write cycle and asserting the data transfer acknowledge signal ($\overline{DTACK}$) to terminate the bus cycle. If no slave responds or the access is invalid, external control logic asserts the $\overline{BERR}$, or $\overline{BERR}$ and $\overline{HALT}$, signal to abort or re-run the bus cycle.

The $\overline{DTACK}$ signal is allowed to be asserted before the data from a slave device is valid on a read cycle. The length of time that $\overline{DTACK}$ may precede data is given as parameter #31 and it must be met in any asynchronous system to insure that valid data is latched into the processor. Notice that there is no maximum time specified from the assertion of $\overline{AS}$ to the assertion of $\overline{DTACK}$. This is because the MPU will insert wait cycles of one clock period each until $\overline{DTACK}$ is recognized.

The $\overline{BERR}$ signal is allowed to be asserted after the $\overline{DTACK}$ signal is asserted. $\overline{BERR}$ must be asserted within the time given as parameter #48 after $\overline{DTACK}$ is asserted in any asynchronous

system to insure proper operation. If this maximum delay time is violated, the processor may exhibit erratic behavior.

Synchronous Operation

To allow for those systems which use the system clock as a signal to generate $\overline{DTACK}$ and other asynchronous inputs, the asynchronous input setup time is given as parameter #47. If this setup is met on an input, such as $\overline{DTACK}$, the processor is guaranteed to recognize that signal on the next falling edge of the system clock. However, the converse is not true – if the input signal does not meet the setup time it is not guaranteed not to be recognized. In addition, if $\overline{DTACK}$ is recognized on a falling edge, valid data will be latched into the processor (on a read cycle) on the next falling edge provided that the data meets the setup time given as parameter #27. Given this, parameter #31 may be ignored. Note that if $\overline{DTACK}$ is asserted, with the required setup time, before the falling edge of S4, no wait status will be incurred and the bus cycle will run at its maximum speed of four clock periods.

In order to assure proper operation in a synchronous system when $\overline{BERR}$ is asserted after $\overline{DTACK}$, $\overline{BERR}$ must meet the setup time parameter #27A prior to the falling edge of the clock one clock cycle after $\overline{DTACK}$ was recognized. This setup time is critical to proper operation, and the HD68000 may exhibit erratic behavior if it is violated.

(NOTE)

During an active bus cycle, $\overline{VPA}$ and $\overline{BERR}$ are sampled on every falling edge of the clock starting with S0. $\overline{DTACK}$ is sampled on every falling edge of the clock starting with S4 and data is latched on the falling edge of S6 during a read. The bus cycle will then be terminated in S7 except when $\overline{BERR}$ is asserted in the absence of $\overline{DTACK}$, in which case it will terminate one clock cycle later in S9.

■ PROCESSING STATES

This section describes the actions the HD68000 which are outside the normal processing associated with the execution of instructions. The functions of the bits in the supervisor portion of the status register are covered: the supervisor/user bit, the trace enable bit, and the processor interrupt priority mask. Finally, the sequence of memory references and actions taken by the processor on exception conditions is detailed.

The HD68000 is always in one of three processing states: normal, exception, or halted. The normal processing state is that associated with instruction execution; the memory references are to fetch instructions and operands, and to store results. A special case of the normal state is the stopped state which the processor enters when a STOP instruction is executed. In this state, no further memory references are made.

The exception processing state is associated with interrupts, trap instructions, tracing and other exceptional conditions. The exception may be internally generated by an instruction or by an unusual condition arising during the execution of an instruction. Externally, exception processing can be forced by an interrupt, by a bus error, or by a reset. Exception processing is designed to provide an efficient context switch so that the processor may handle unusual conditions.

The halted processing state is an indication of catastrophic hardware failure. For example, if during the exception processing of a bus error another bus error occurs, the processor assumes that the system is unusable and halts. Only an external reset can restart a halted processor. Note that a processor in the stopped state is not in the halted state, nor vice versa.

PROCESSING STATES

NORMAL	INSTRUCTION EXECUTION (INCLUDING STOP)
EXCEPTION	INTERRUPTS TRAPS TRACING ETC.
HALTED	HARDWARE HALT DOUBLE BUS FAULT

• PRIVILEGE STATES

The processor operates in one of two states of privilege: the "user" state or the "supervisor" state. The privilege state determines which operations are legal, are used to choose between the supervisor stack pointer and the user stack pointer in instruction references, and may be used by an external memory management device to control and translate accesses.

The privileges state is a mechanism for providing security in a computer system. Programs should access only their own code and data areas, and ought to be restricted from accessing information which they do not need and must not modify.

The privilege mechanism provides security by allowing most programs to execute in user state. In this state, the accesses are controlled, and the effects on other parts of the system are limited. The operating system executes in the supervisor state, has access to all resources, and performs the overhead tasks for the user state programs.

SUPERVISOR STATE

The supervisor state is the higher state of privilege. For instruction execution, the supervisor state is determined by the S-bit of the status register; if the S-bit is asserted (high), the processor is in the supervisor state. All instructions can be executed in the supervisor state. The bus cycles generated by instructions executed in the supervisor state are classified as supervisor references. While the processor is in the supervisor privilege state, those instructions which use either the system stack pointer implicitly or address register seven explicitly access the supervisor stack pointer.

All exception processing is done in the supervisor state, regardless of the setting of the S-bit. The bus cycles generated during exception processing are classified as supervisor references. All stacking operations during exception processing use the supervisor stack pointer.

USER STATE

The user state is the lower state of privilege. For instruction execution, the user state is determined by the S-bit of the status register; if the S-bit is negated (low), the processor is executing instructions in the user state.

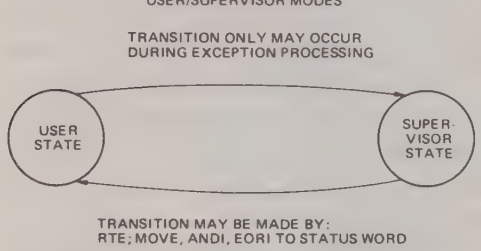
Most instructions execute the same in user state as in the supervisor state. However, some instructions which have important system effects are made privileged. User programs are not permitted to execute the STOP instruction, or the

RESET instruction. To ensure that a user program cannot enter the supervisor state except in a controlled manner, the instructions which modify the whole status register are privileged. To aid in debugging programs which are to be used as operating systems, the move to user stack pointer (MOVE to USP) and move from user stack pointer (MOVE from USP) instructions are also privileged.

The bus cycles generated by an instruction executed in user state are classified as user state references. This allows an external memory management device to translate the address and to control access to protected portions of the address space. While the processor is in the user privilege state, those instructions which use either the system stack pointer implicitly, or address register seven explicitly, access the use stack pointer.

PRIVILEGE STATE CHANGES

Once the processor is in the user state and executing instructions, only exception processing can change the privilege state. During exception processing, the current setting of the S-bit of the status register is saved and the S-bit is asserted, putting the processing in the supervisor state. Therefore, when instruction execution resumes at the address specified to process the exception, the processor is in the supervisor privilege state.



REFERENCE CLASSIFICATION

When the processor makes a reference, it classifies the kind of reference being made, using the encoding on the three function code output lines. This allows external translation of addresses, control of access, and differentiation of special processor states, such as interrupt acknowledge. Table 18 lists the classification of references.

Table 18 Reference Classification

Function Code Output			Reference Class
FC_2	FC_1	FC_0	
0	0	0	(Unassigned)
0	0	1	User Data
0	1	0	User Program
0	1	1	(Unassigned)
1	0	0	(Unassigned)
1	0	1	Supervisor Data
1	1	0	Supervisor Program
1	1	1	Interrupt Acknowledge

• EXCEPTION PROCESSING

Before discussing the details of interrupts, traps, and tracing, a general description of exception processing is in order. The processing of an exception occurs in four steps, with variations for different exception causes. During the first step, a temporary copy of the status register is made, and the status register is set for exception processing. In the second step the exception vector is determined, and the third step is the saving of the current processor context. In the fourth step a new context is obtained, and the processor switches to instruction processing.

EXCEPTION VECTORS

Exception vectors are memory locations from which the processor fetches the address of a routine which will handle that exception. All exception vectors are two words in length (Figure 34), except for the reset vector, which is four words. All exception vectors lie in the supervisor data space, except for the reset vector which is in the supervisor program space. A vector number is an eight-bit number which, when multiplied by four, gives the address of an exception vector. Vector numbers are generated internally or externally depending on the cause of the exception. In the case of interrupts, during the interrupt acknowledge bus cycle, a peripheral provides an 8-bit vector number (Figure 35) to the processor on data bus lines D_0 through D_7. The processor translates the vector number into a full 24-bit address, as shown in Figure 36. The memory layout for exception vectors is given in Table 19.

As shown in Table 19, the memory layout is 512 words long (1024 bytes). It starts at address 0 and proceeds through address 1023. This provides 255 unique vectors; some of these are reserved for TRAPS and other system functions. Of the 255, there are 192 reserved for user interrupt vectors. However, there is no protection on the first 64 entries, so user interrupt vectors may overlap at the discretion of the systems designer.

KINDS OF EXCEPTIONS

Exceptions can be generated by either internal or external causes. The externally generated exceptions are the interrupts and the bus error and reset requests. The interrupts are requests from peripheral devices for processor action while the bus error and reset inputs are used for access control and processor restart. The internally generated exceptions come from instructions, or from address error or tracing. The trap (TRAP), trap on overflow (TRAPV), check register against bounds (CHK) and divide (DIV) instructions all can generate exceptions as part of their instruction execution. In addition, illegal instructions, word fetches from odd addresses and privilege violations cause exceptions. Tracing behaves like a very high priority, internally generated interrupt after each instruction execution.

EXCEPTION PROCESSING SEQUENCE

Exception processing occurs in four identifiable steps. In the first step, an internal copy is made of the status register. After the copy is made, the S-bit is asserted, putting the processor into the supervisor privilege state. Also, the T-bit is negated which will allow the exception handler to execute unhindered by tracing. For the reset and interrupt exceptions, the interrupt priority mask is also updated.

In the second step, the vector number of the exception is determined. For interrupts, the vector number is obtained by a processor fetch, classified as an interrupt acknowledge. For all other exceptions, internal logic provides the vector number. This vector number is then used to generate the address of the exception vector.

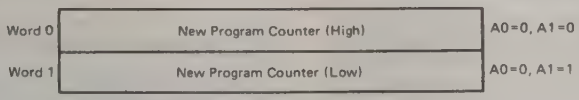
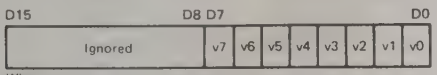


Figure 34 Exception Vector Format



Where
v7 is the MSB of the Vector Number
v0 is the LSB of the Vector Number

Figure 35 Peripheral Vector Number Format

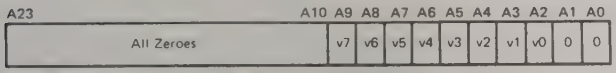


Figure 36 Address Translated From 8-Bit Vector Number

Table 19 Exception Vector Assignment

Vector Number(s)	Address			Assignment
	Dec	Hex	Space	
0	0	000	SP	Reset: Initial SSP
–	4	004	SP	Reset: Initial PC
2	8	008	SD	Bus Error
3	12	00C	SD	Address Error
4	16	010	SD	Illegal Instruction
5	20	014	SD	Zero Divide
6	24	018	SD	CHK Instruction
7	28	01C	SD	TRAPV Instruction
8	32	020	SD	Privilege Violation
9	36	024	SD	Trace
10	40	028	SD	Line 1010 Emulator
11	44	02C	SD	Line 1111 Emulator
12*	48	030	SD	(Unassigned, reserved)
13*	52	034	SD	(Unassigned, reserved)
14*	56	038	SD	(Unassigned, reserved)
15	60	03C	SD	Uninitialized Interrupt Vector
16 ~ 23*	64	040	SD	(Unassigned, reserved)
	95	05F		
24	96	060	SD	Spurious Interrupt
25	100	064	SD	Level 1 Interrupt Autovector
26	104	068	SD	Level 2 Interrupt Autovector
27	108	06C	SD	Level 3 Interrupt Autovector
28	112	070	SD	Level 4 Interrupt Autovector
29	116	074	SD	Level 5 Interrupt Autovector
30	120	078	SD	Level 6 Interrupt Autovector
31	124	07C	SD	Level 7 Interrupt Autovector
32 ~ 47	128	080	SD	TRAP Instruction Vectors
	191	0BF		
48 ~ 63*	192	0C0	SD	(Unassigned, reserved)
	255	0FF		
64 ~ 255	256	100	SD	User Interrupt Vectors
	1023	3FF		

SP: Supervisor program, SD: Supervisor data

* Vector numbers 12, 13, 14, 16 through 23 and 48 through 63 are reserved for future enhancements by Hitachi. No user peripheral devices should be assigned these numbers

The third step is to save the current processor status, except for the reset exception. The current program counter value and the saved copy of the status register are stacked using the supervisor stack pointer as shown in Figure 37. The program counter value stacked usually points to the next unexecuted instruction, however for bus error and address error, the value stacked for the program counter is unpredictable, and may be incremented from the address of the instruction which caused the error. Additional information defining the current context is stacked for the bus error and address error exceptions.

The last step is the same for all exceptions. The new program counter value is fetched from the exception vector. The processor then resumes instruction execution. Then instruction at the address given in the exception vector is fetched, and normal instruction decoding and execution is started.

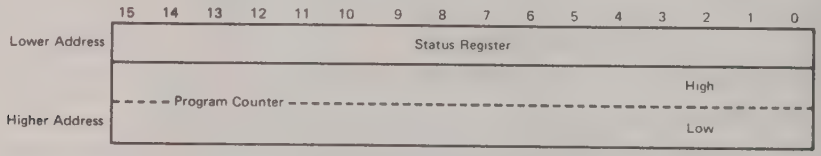


Figure 37 Exception Stack Order (Group 1, 2)

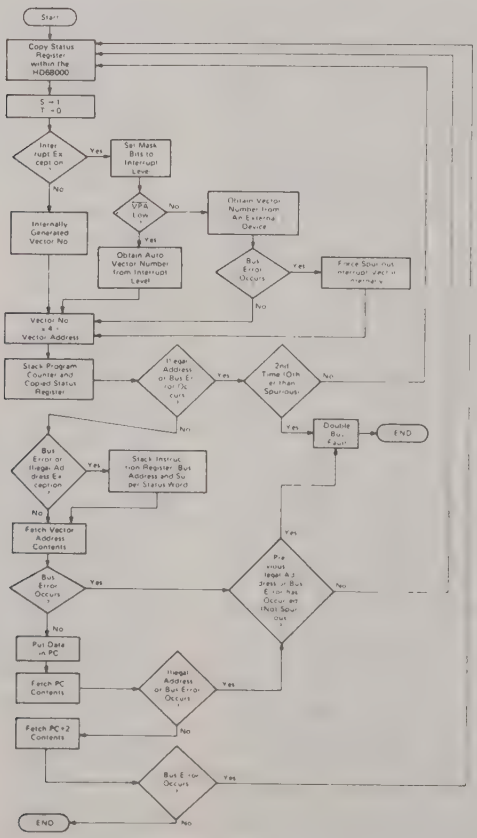


Figure 38 Exception Processing Sequence (Not Reset)

MULTIPLE EXCEPTIONS

These paragraphs describe the processing which occurs when multiple exceptions arise simultaneously. Exceptions can be grouped according to their occurrence and priority. The Group 0 exceptions are reset, bus error, and address error. These exceptions cause the instruction currently being executed to be aborted, and the exeception processing to commence within two clock cycles. The Group 1 exceptions are trace and interrupt, as well as the privilege violations and illegal instructions. These exceptions allow the current instruction to execute to completion, but preempt the execution of the next instruction by forcing exception processing to occur (privilege violations and illegal instructions are detected when they are the next instruction to be executed). The Group 2 exceptions occur as part of the normal processing of instructions. The TRAP, TRAPV, CHK, and zero divide exceptions are in this group. For these exceptions, the normal execution of an instruction may lead to exception processing.

Group 0 exceptions have highest priority, while Group 2 exceptions have lowest priority. Within Group 0, reset has highest priority, followed by address error and then bus error. Within Group 1, trace has priority over external interrupts, which in turn takes priority over illegal instruction and privilege violation. Since only one instruction can be executed at a time, there is no priority relation within Group 2.

The priority relation between two exceptions determines which is taken, or taken first, if the conditions for both arise simultaneously. Therefore, if a bus error occurs during a TRAP instruction, the bus error takes precedence, and the TRAP instruction processing is aborted. In another example, if an interrupt request occurs during the execution of an instruction while the T-bit is asserted, the trace exception has priority, and is processed first. Before instruction processing resumes, however, the interrupt exception is also processed, and instruction processing commences finally in the interrupt handler routine. A summary of exception grouping and priority is given in Table 20.

Table 20 Exception Grouping and Priority

Group	Exception	Processing
0	Reset Address Error Bus Error	Exception processing begins within two clock cycles.
1	Trace Interrupt Illegal Privilege	Exception processing begins before the next instruction
2	TRAP, TRAPV CHK, Zero Divide	Exception processing is started by normal instruction execution

RECOGNITION TIMES OF EXCEPTIONS, HALT, AND BUS ARBITRATION

END OF A CLOCK CYCLE
- RESET

END OF A BUS CYCLE
- ADDRESS ERROR
- BUS ERROR
- HALT
- BUS ARBITRATION

END OF AN INSTRUCTION CYCLE
- TRACE EXCEPTION
- INTERRUPT EXCEPTIONS
- ILLEGAL INSTRUCTION
- UNIMPLEMENTED INSTRUCTION
- PRIVILEGE VIOLATION

WITHIN AN INSTRUCTION CYCLE
- TRAP, TRAPV
- CHK
- ZERO DIVIDE

• EXCEPTION PROCESSING DETAILED DISCUSSION

Exceptions have a number of sources, and each exception has processing which is peculiar to it. The following paragraphs detail the sources of exceptions, how each arises, and how each is processed.

RESET

The reset input provides the highest exception level. The processing of the reset signal is designed for system initiation, and recovery from catastrophic failure. Any processing in progress at the time of the reset is aborted and cannot be recovered. The processor is forced into the supervisor state, and the trace state is forced off. The processor interrupt priority mask is set at level seven. The vector number is internally generated to reference the reset exception vector at location 0 in the supervisor program space. Because no assumptions can be made about the validity of register contents, in particular the supervisor stack pointer, neither the program counter nor the status register is saved. The address contained in the first two words of the reset exception vector is fetched as the initial supervisor stack pointer, and the address in the last two words of the reset exception vector is fetched as the initial program counter. Finally, instruction execution is started at the address in the program counter. The power-up/restart code should be pointed to by the initial program counter.

The RESET instruction does not cause loading of the reset vector, but does assert the reset line to reset external devices. This allows the software to reset the system to a known state and then continue processing with the next instruction.

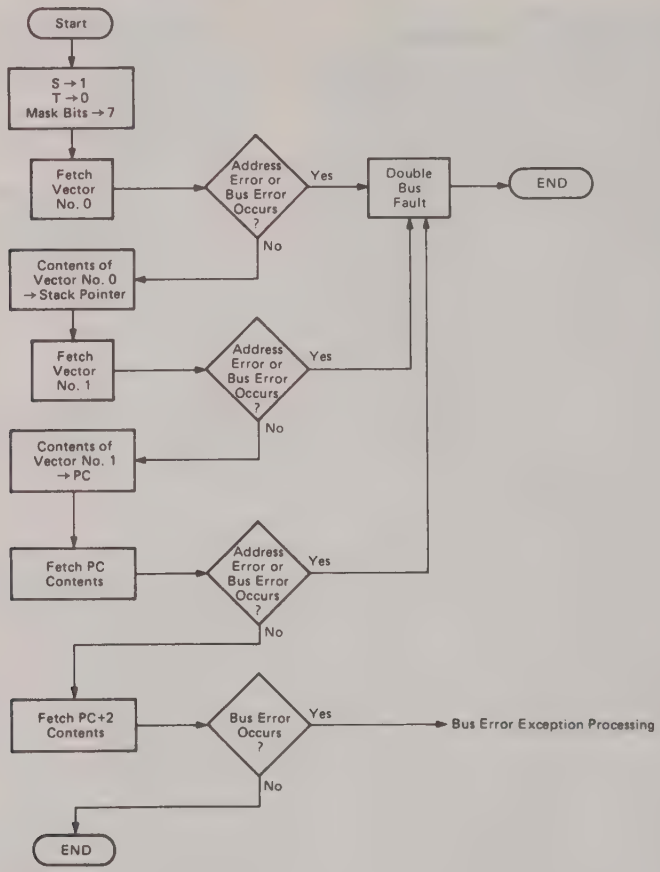


Figure 39 Reset Exception Processing

INTERRUPTS

Seven levels of interrupt priorities are provided. Devices may be chained externally within interrupt priority levels, allowing an unlimited number of peripheral devices to interrupt the processor. Interrupt priority levels are numbered from one to seven, with level seven being the highest priority. The status register contains a three-bit mask which indicates the current processor priority, and interrupts are inhibited for all priority levels less than or equal to the current processor priority.

An interrupt request is made to the processor by encoding the interrupt request level on the interrupt request lines; a zero indicates no interrupt request. Interrupt requests arriving at the processor do not force immediate exception processing, but are made pending. Pending interrupts are detected between instruction executions. If the priority of the pending interrupt is lower than or equal to the current processor priority, execution continues with the next instruction and the interrupt exception processing is postponed. (The recognition of level seven is slightly different, as explained in a following paragraph.)

If the priority of the pending interrupt is greater than the current processor priority, the exception processing sequence is started. First a copy of the status register is saved, and the privilege state is set to supervisor, tracing is suppressed, and the processor priority level is set to the level of the interrupt being acknowledged. The processor fetches the vector number from the interrupting device, classifying the reference as an interrupt acknowledge and displaying the level number of

the interrupt being acknowledged on the address bus. If external logic requests an automatic vectoring, the processor internally generates a vector number which is determined by the interrupt level number. If external logic indicates a bus error, the interrupt is taken to be spurious, and the generated vector number references the spurious interrupt vector. The processor then proceeds with the usual exception processing, saving the program counter and status register on the supervisor stack. The saved value of the program counter is the address of the instruction which would have been executed had the interrupt not been present. The content of the interrupt vector whose vector number was previously obtained is fetched and loaded into the program counter, and normal instruction execution commences in the interrupt handling routine. A flow chart for the interrupt acknowledge sequence is given in Figure 40, a timing diagram is given in Figure 41, and the interrupt exception timing sequence is shown in Figure 42.

Table 21 Internal Interrupt Level

Level	I2	I1	I0	Interrupt
7	1	1	1	Non-Maskable Interrupt
6	1	1	0	Maskable Interrupt
5	1	0	1	
4	1	0	0	
3	0	1	1	
2	0	1	0	
1	0	0	1	
0	0	0	0	No Interrupt

(NOTE) The internal interrupt mask level (I2, I1, I0) are inverted to the logic level applied to the pins ($\overline{IPL_2}$, $\overline{IPL_1}$, $\overline{IPL_0}$).

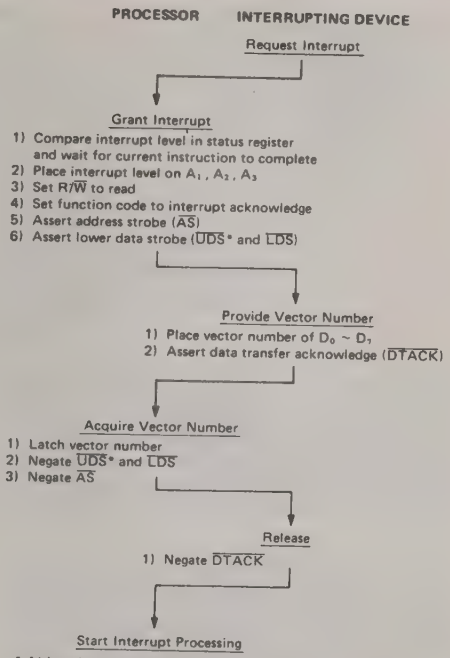
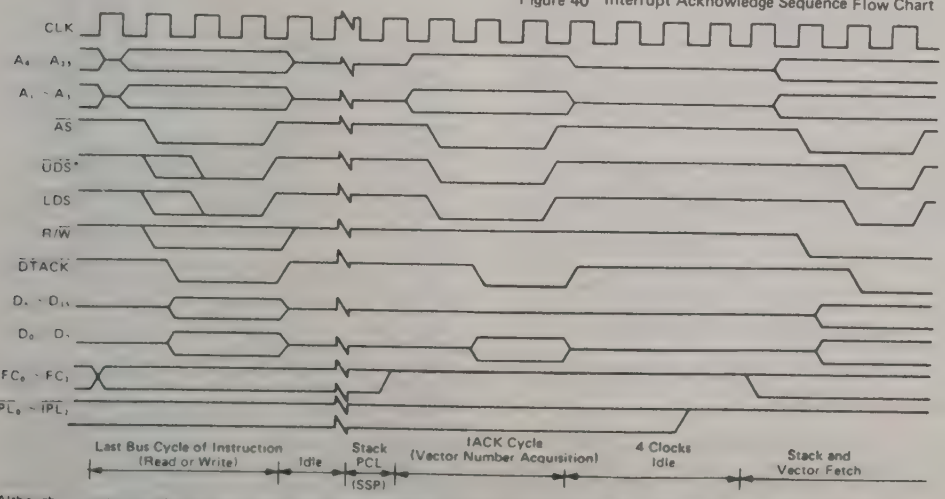
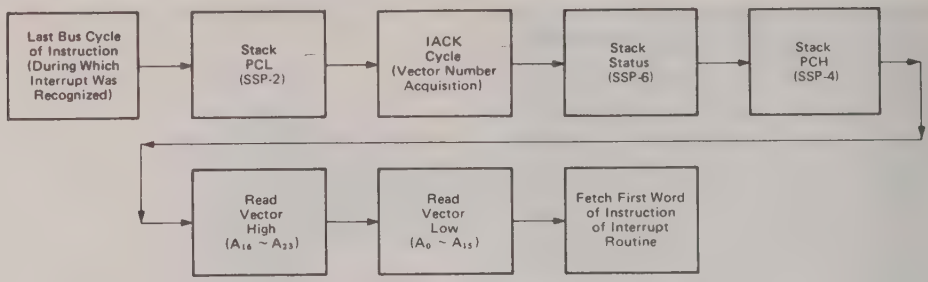

PROCESSOR
INTERRUPTING DEVICE
Request Interrupt
Grant Interrupt
1) Compare interrupt level in status register and wait for current instruction to complete
2) Place interrupt level on A1, A2, A3
3) Set R/W to read
4) Set function code to interrupt acknowledge
5) Assert address strobe (AS)
6) Assert lower data strobe (UDS* and LDS)
Provide Vector Number
1) Place vector number of D0 ~ D7
2) Assert data transfer acknowledge (DTACK)
Acquire Vector Number
1) Latch vector number
2) Negate UDS* and LDS
3) Negate AS
Release
1) Negate DTACK
Start Interrupt Processing
* Although a vector number is one byte, both data strobes are asserted due to the microcode used for exception processing. The processor does not recognize anything on data lines D8 through D15 at this time.

Figure 40 Interrupt Acknowledge Sequence Flow Chart



* Although a vector number is one byte, both data strobes are asserted due to the microcode used for exception processing. The processor does not recognize anything on data lines D_8 through D_{15} at this time.

Figure 41 Interrupt Acknowledge Sequence Timing Diagram


Last Bus Cycle of Instruction (During Which Interrupt Was Recognized)
Stack PCL (SSP-2)
IACK Cycle (Vector Number Acquisition)
Stack Status (SSP-6)
Stack PCH (SSP-4)
Read Vector High (A_{16} ~ A_{23})
Read Vector Low (A_0 ~ A_{15})
Fetch First Word of Instruction of Interrupt Routine

Note: SSP refers to the value of the supervisor stack pointer before the interrupt occurs.

Figure 42 Interrupt Exception Timing Sequence

Priority level seven is a special case. Level seven interrupts cannot be inhibited by the interrupt priority mask, thus providing a "non-maskable interrupt" capability. An interrupt is generated each time the interrupt request level changes from some lower level to level seven. Note that a level seven interrupt may still be caused by the level comparison if the request level is a seven and the processor priority is set to a lower level by an instruction.

UNINITIALIZED INTERRUPT

An interrupting device asserts $\overline{VPA}$ or provides an interrupt vector during an interrupt acknowledge cycle to the HD68000. If the vector register has not been initialized, the responding HMCS68000 Family peripheral will provide vector 15, the unitialized interrupt vector. This provides a uniform way to recover from a programming error.

SPURIOUS INTERRUPT

If during the interrupt acknowledge cycle no device responds by asserting $\overline{DTACK}$ or $\overline{VPA}$, the bus error line should be asserted to terminate the vector acquisition. The processor separates the processing of this error from bus error by fetching the spurious interrupt vector instead of the bus error vector. The processor then proceeds with the usual exception processing.

INSTRUCTION TRAPS

Traps are exceptions caused by instructions. They arise either from processor recognition of abnormal conditions during instruction execution, or from use of instructions whose normal behavior is trapping.

Some instructions are used specifically to generate traps. The TRAP instruction always forces an exception, and is useful for implementing system calls for user programs. The TRAPV and CHK instructions force an exception if the user program detects a runtime error, which may be an arithmetic overflow or a subscript out of bounds.

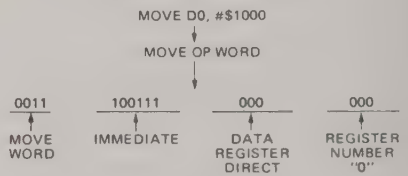
The signed divide (DIVS) and unsigned divide (DIVU) instructions will force an exception if a division operation is attempted with a divisor of zero.

ILLEGAL AND UNIMPLEMENTED INSTRUCTIONS

Illegal instruction is the term used to refer to any of the word bit patterns which are not the bit pattern of the first word of a legal instruction. During instruction execution, if such an instruction is fetched, an illegal instruction exception occurs.

Word patterns with bits 15 through 12 equaling 1010 or 1111 are distinguished as unimplemented instructions and separate exception vectors are given to these patterns to permit efficient emulation. This facility allows the operating system to detect program errors, or to emulate unimplemented instructions in software.

ILLEGAL INSTRUCTION EXAMPLE



PRIVILEGE VIOLATIONS

In order to provide system security, various instructions are privileged. An attempt to execute one of the privileged instructions while in the user state will cause an exception. The privileged instruction are:

STOP	AND (word) Immediate to SR
RESET	EOR (word) Immediate to SR
RTE	OR (word) Immediate to SR
MOVE to SR	MOVE USP

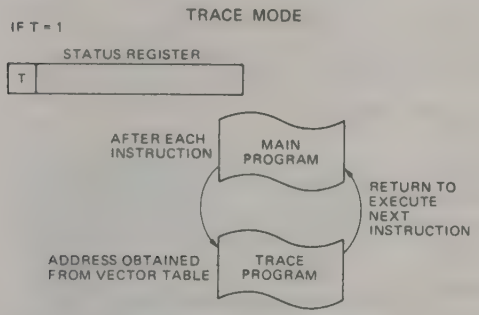
TRACING

To aid in program development, the HD68000 includes a facility to allow instruction by instruction tracing. In the trace state, after each instruction is executed an exceptions is forced, allowing a debugging program to monitor the execution of the program under test.

The trace facility uses the T-bit in the supervisor portion of the status register. If the T-bit is negated (off), tracing is disabled, and instruction execution proceeds from instruction to instruction as normal. If the T-bit is asserted (on) at the beginning of the execution of an instruction, a trace exception will be generated after the execution of that instruction is completed. If the instruction is not executed. either because an interrupt is taken, or the instruction is illegal or privileged, the trace exception does not occur. The trace exception also does not occur if the instruction is aborted by a reset, bus

error, or address error exception. If the instruction is indeed executed and an interrupt is pending on completion, the trace exception is processed before the interrupt exception. If, during the execution of the instruction, an exception is forced by that instruction, the forced exception is processed before the trace exception.

As an extreme illustration of the above rules, consider the arrival of an interrupt during the execution of a TRAP instruction while tracing is enabled. First the trap exception is processed, then the trace exception, and finally the interrupt exception. Instruction execution resumes in the interrupt handler routine.



1. If, upon completion of an instruction, T = 1, go to trace exception processing.
2. Execute trace exception sequence.
3. Execute trace service routine.
4. At the end of the service routine, execute return from exception (RTE).

BUS ERROR

Bus error exceptions occur when the external logic requests that a bus error be processed by an exception. The current bus cycle which the processor is making is then aborted. Whether the processor was doing instruction or exception processing, that processing is terminated, and the processor immediately begins exception processing.

Exception processing for bus error follows the usual sequence of steps. The status register is copied, the supervisor state is entered, and the trace state is turned off. The vector number is generated to refer to the bus error vector. Since the processor was not between instructions when the bus error exception request was made, the context of the processor is more detailed. To save more of this context, additional information is saved on the supervisor stack. The program counter and the copy of the status register are of course saved. The value saved for the program counter is advanced by some amount, one to five words beyond the address of the first word of the instruction which made the reference causing the bus error. If the bus error occurred during the fetch of the next instruction, the saved program counter has a value in the vicinity of the current instruction, even if the current instruction is a branch, a jump, or a return instruction. Besides the usual information, the processor saves its internal copy of the first word of the instruction being processed, and the address which was being accessed by the aborted bus cycle. Specific information about the access is also saved: whether it was a read or a write, whether the processor was processing an instruction or not, and the classification displayed on the function code outputs when the bus error occurred. The processor is processing an instruction if it is in the normal state or processing a Group 2 exception; the processor is not processing an instruction if it is processing a Group 0 or a Group 1 exception. Figure 43 illustrates how this information is organized on the supervisor stack. Although this information is not sufficient in general to effect full recovery from the bus error, it does allow software diagnosis. Finally, the processor commences instruction processing at the address contained in the vector. It is the responsibility of the error handler routine to clean up the stack and determine where to continue execution.

If a bus error occurs during the exception processing for a bus error, address error, or reset, the processor is halted, and all processing cases. This simplifies the detection of catastrophic system failure, since the processor removes itself from the system rather than destroy all memory contents. Only the $\overline{RES}$ pin can restart a halted processor.

ADDRESS ERROR

Address error exceptions occur when the processor attempts to access a word or a long word operand or an instruction at an odd address. The effect is much like an internally generated bus error, so that the bus cycle is aborted, and the processor ceases whatever processing it is currently doing and begins exception processing. After exception processing commences, the sequence is the same as that for bus error including the information that is stacked, except that the vector number refers to the address error vector instead. Likewise, if an address error occurs during the exception processing for a bus error, address error, or reset, the processor is halted. As shown in Figure 44, an address error will execute a short bus cycle followed by exception processing.

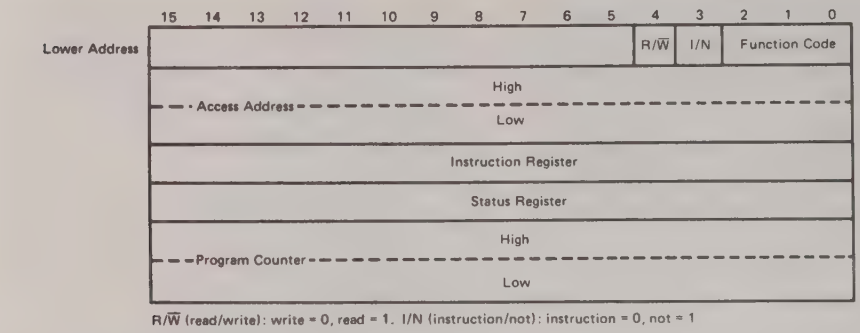


Figure 43 Exception Stack Order (Group 0)

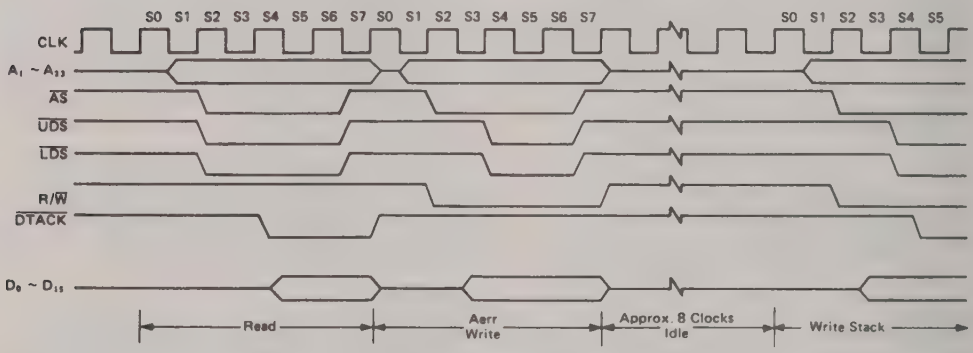


Figure 44 Address Error Timing

■ INTERFACE WITH HMCS6800 PERIPHERALS

Hitachi's extensive line of HMCS6800 peripherals are directly compatible with the HD68000. Some of these devices that are particularly useful are:

HD6821	Peripheral Interface Adapter
HD6840	Programmable Timer Module
HD6843	Floppy Disk Controller
HD6845S	CRT Controller
HD46508	Analog Data Acquisition Unit
HD6850	Asynchronous Communication Interface Adapter
HD6852	Synchronous Serial Data Adapter

To interface the synchronous HMCS6800 peripherals with the asynchronous HD68000, the processor modifies its bus cycle to meet the HMCS6800 cycle requirements whenever an HMCS6800 device address is detected. This is possible since both processors use memory mapped I/O. Figure 46 is a flow chart of the interface operation between the processor and HMCS6800 devices.

● DATA TRANSFER OPERATION

Three signal on the processor provide the HMCS6800 interface. They are: enable (E), valid memory address ($\overline{\text{VMA}}$), and valid peripheral address ($\overline{\text{VPA}}$). Enable corresponds to the E or ϕ_2 signal in existing HMCS6800 systems. The bus frequency is one tenth of the incoming HD68000 clock frequency. The timing of E allows 1 MHz peripherals to be used with an 8 MHz HD68000. Enable has a 60/40 duty cycle; that is, it is low for six input clocks and high for four input clocks. This duty cycle allows the processor to do successive $\overline{\text{VPA}}$ accesses on successive E pulses.

HMCS6800 cycle timing is given in Figures 47 and 48. At state zero (S0) in the cycle, the address bus is in the high-impedance state. A function code is asserted on the function code output lines. One-half clock later, in state 1 the address bus is released from the high-impedance state.

During state 2, the address strobe ($\overline{\text{AS}}$) is asserted to indicate that there is a valid address on the address bus. If the bus cycle is a read cycle, the upper and/or lower data strobes are also asserted in state 2. If the bus cycle is a write cycle,

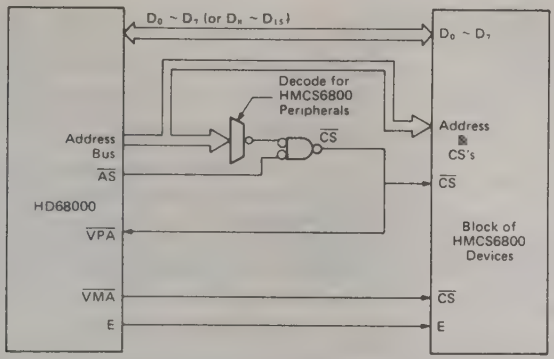


Figure 45 Connection of HMCS6800 Peripherals

the read/write (R/$\overline{W}$) signal is switched to low (write) during state 2. One half clock later, in state 3, the write data is placed on the data bus, and in state 4 the data strobes are issued to indicate valid data on the data bus. The processor now inserts wait states until it recognizes the assertion of $\overline{VPA}$.

The $\overline{VPA}$ input signals the processor that the address on the bus is the address of an HMCS6800 device (or an area reserved for HMCS6800 devices) and that the bus should conform to the ϕ_2 transfer characteristics of the HMCS6800 bus. Valid peripheral address is derived by decoding the address bus, conditioned by address strobe. Chip select for the HMCS6800 peripherals should be derived by decoding the address bus conditioned by $\overline{VMA}$.

After the recognition of $\overline{VPA}$, the processor assures that the Enable (E) is low, by waiting if necessary, and subsequently asserts $\overline{VMA}$. Valid memory address is then used as part of the chip select equation of the peripheral. This ensures that the HMCS6800 peripherals are selected and deselected at the correct time. The peripheral now runs its cycle during the high portion of the E signal. Figures 47 and 48 depict the best and worst case HMCS6800 cycle timing. This cycle length is dependent strictly upon when $\overline{VPA}$ is asserted in relationship to the E clock.

If we assume that external circuitry asserts $\overline{VPA}$ as soon as possible after the assertion of $\overline{AS}$, then $\overline{VPA}$ will be recognized as being asserted on the falling edge of S4. In this case, no "extra" wait cycles will be inserted prior to the recognition of $\overline{VPA}$ asserted and only the wait cycles inserted to synchronize with the E clock will determine the total length of the cycle. In any case, the synchronization delay will be some integral number of clock cycles within the following two extremes:

1. Best Case – $\overline{VPA}$ is recognized as being asserted on the falling edge three clock cycles before E rises (or three clock cycles after E falls).
2. Worst Case – $\overline{VPA}$ is recognized as being asserted on the falling edge two clock cycles before E rises(or four clock cycles after E falls).

During a read cycle, the processor latches the peripheral data in state 6. For all cycles, the processor negates the address and data strobes one half clock cycle later in state 7, and the Enable signal goes low at this time. Another half clock later, the address bus is put in the high-impedance state. During a write cycle, the data bus is put in the high-impedance state

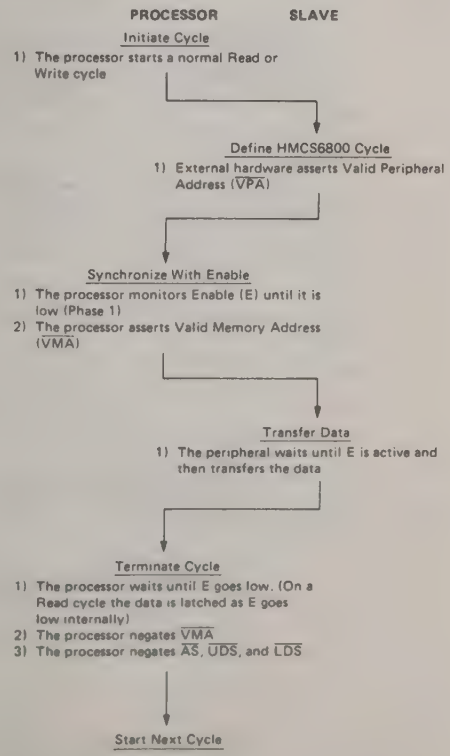

PROCESSOR
SLAVE
Initiate Cycle
1) The processor starts a normal Read or Write cycle
Define HMCS6800 Cycle
1) External hardware asserts Valid Peripheral Address (VPA)
Synchronize With Enable
1) The processor monitors Enable (E) until it is low (Phase 1)
2) The processor asserts Valid Memory Address (VMA)
Transfer Data
1) The peripheral waits until E is active and then transfers the data
Terminate Cycle
1) The processor waits until E goes low. (On a Read cycle the data is latched as E goes low internally)
2) The processor negates VMA
3) The processor negates AS, UDS, and LDS
Start Next Cycle

Figure 46 HMCS6800 Interface Flow Chart

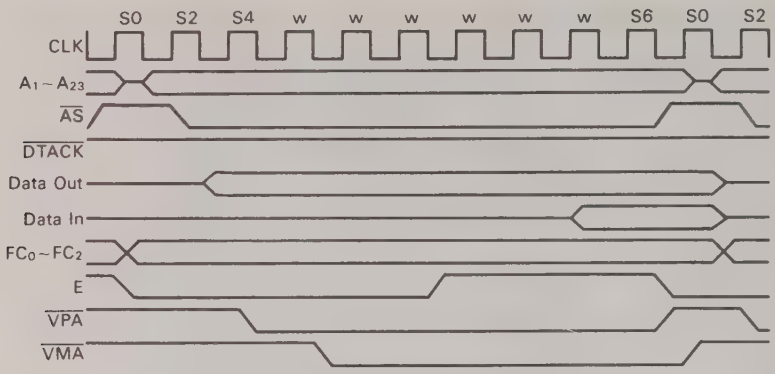


Figure 47 HD68000 to HMCS6800 Peripheral Timing — Best Case

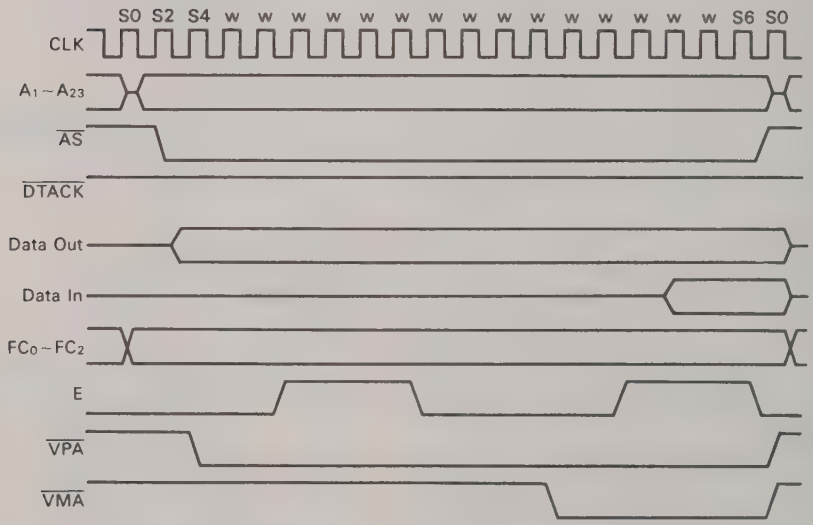


Figure 48 HD68000 to HMCS6800 Peripheral Timing — Worst Case

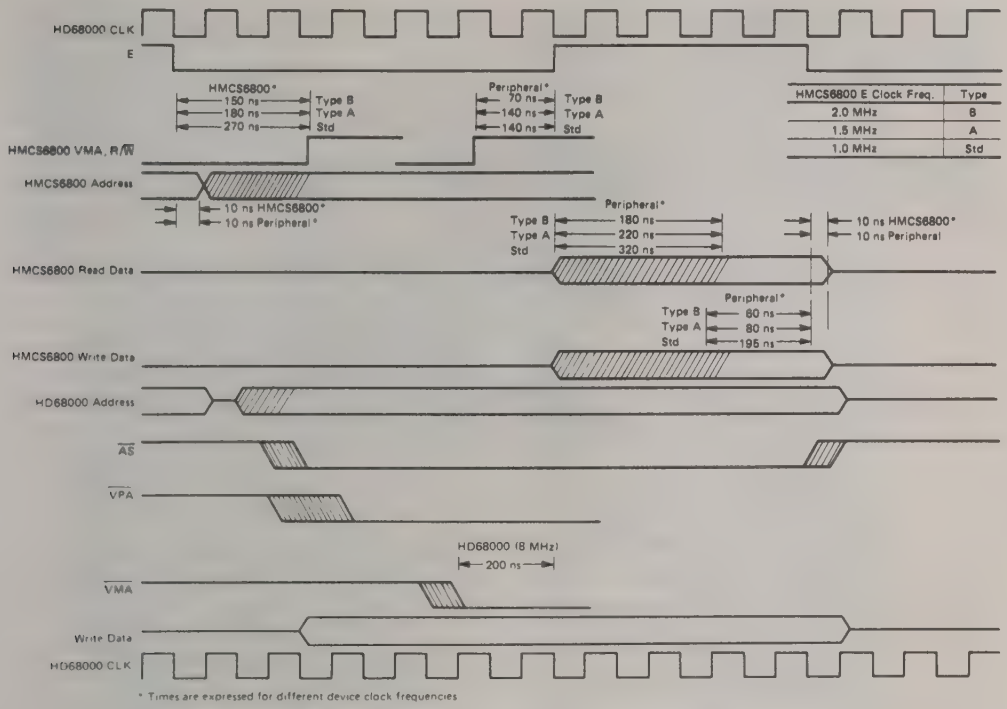


Figure 49 HD68000 to HMCS6800 Peripheral Timing Diagram

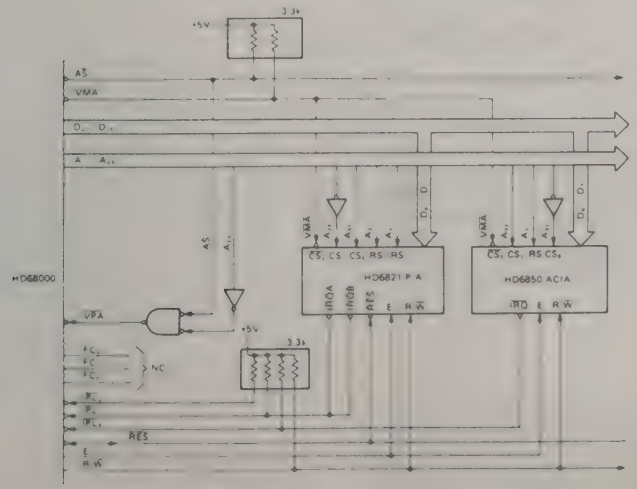


Figure 50 HMCS6800 Interface — Example 1

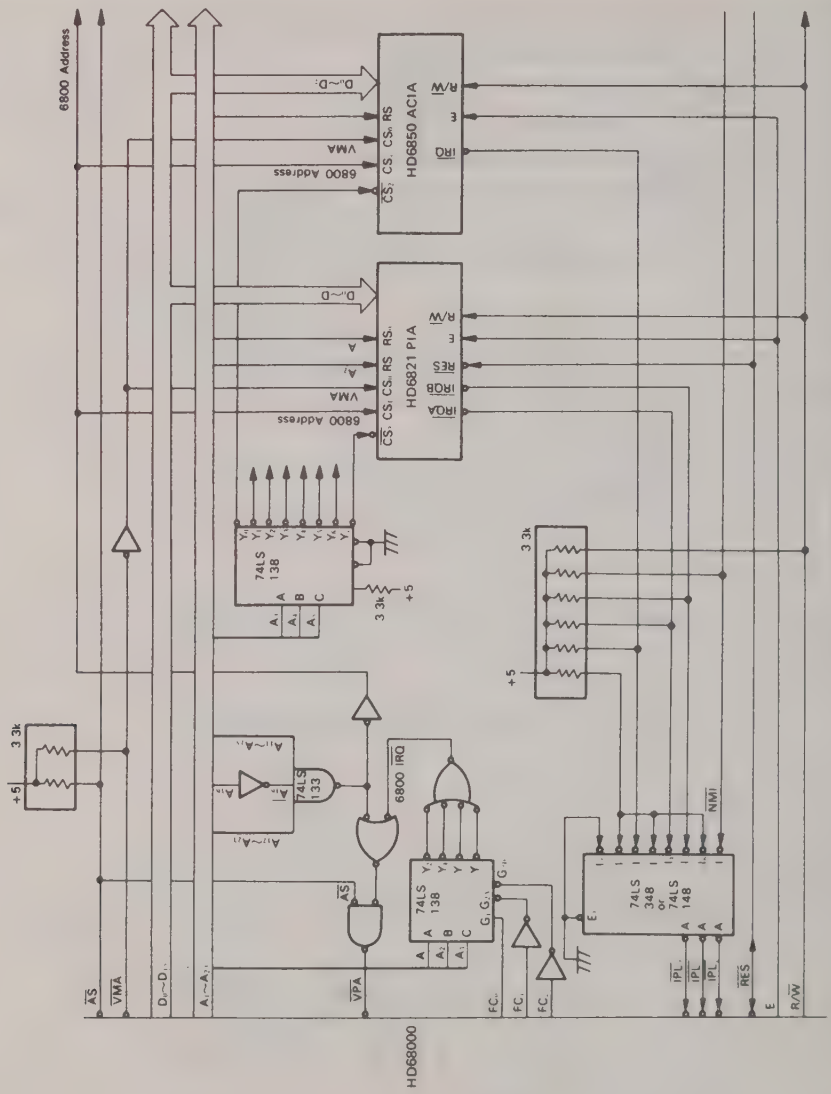


Figure 51 HMCS6800 Interface — Example 2

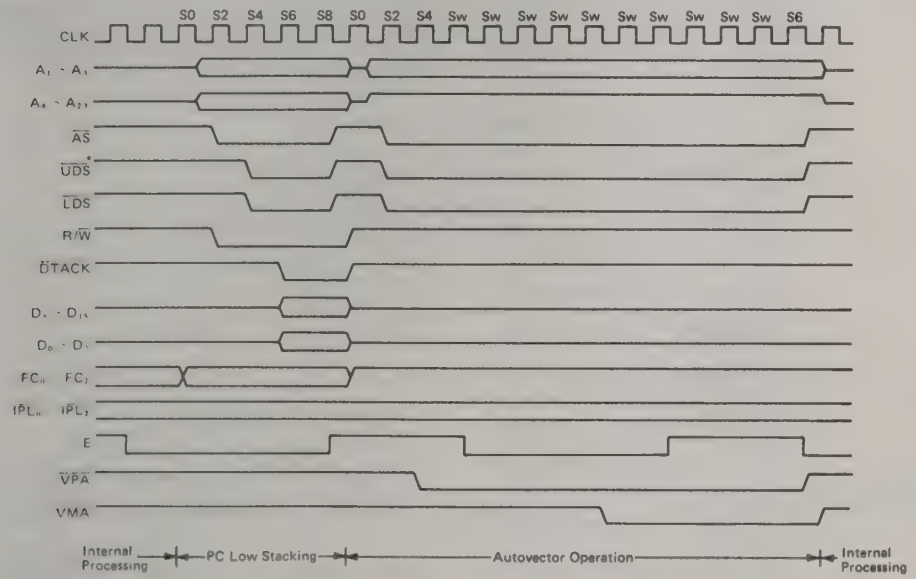
and the read/write signal is switched high. The peripheral logic must remove $\overline{VPA}$ within one clock after address strobe is negated.

Figure 49 shows the timing required by HMCS6800 peripherals, the timing specified for HMCS6800, and the corresponding timing for the HD68000. Two example systems with HMCS6800 peripherals are shown in Figures 50 and 51. The system in Figure 50 reserves the upper eight megabytes of memory for HMCS6800 peripherals. The system in Figure 51 is more efficient with memory and easily expandable, but more complex.

$\overline{DTACK}$ should not be asserted while $\overline{VPA}$ is asserted. Notice that the HD68000 $\overline{VMA}$ is active low, contrasted with the active high HMCS6800 VMA. This allows the processor to put its buses in the high-impedance state on DMA requests without inadvertently selecting peripherals.

• INTERRUPT OPERATION

During an interrupt acknowledge cycle while the processor is fetching the vector, if $\overline{VPA}$ is asserted, the HD68000 will assert $\overline{VMA}$ and complete a normal HMCS6800 read cycle as shown in Figure 52. The processor will then use an internally



* Although a vector number is one byte, both data strobes are asserted due to the microcode used for exception processing. The processor does not recognize anything on data lines D_8 through D_{15} at this time.

Figure 52 Autovector Operation Timing Diagram

generated vector that is a function of the interrupt being serviced. This process is known as autovectoring. The seven autovectors are vector numbers 25 through 31 (decimal).

This operates in the same fashion (but is not restricted to) the HMCS6800 interrupt sequence. The basic difference is that there are six normal interrupt vectors and one NMI type vector. As with both the HMCS6800 and the HD68000's normal vectored interrupt, the interrupt service routine can be located anywhere in the address space. This is due to the fact that while the vector numbers are fixed, the contents of the vector table entries are assigned by the user.

Since $\overline{VMA}$ is asserted during autovectoring, the HMCS6800 peripheral address decoding should prevent unintended accesses.

■ CONDITION CODES COMPUTATION

This provides a discussion of how the condition codes were developed, the meanings of each bit, how they are computed, and how they are represented in the instruction set details.

• CONDITION CODE REGISTER

The condition code register portion of the status register contains five bits:

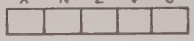
N – Negative
Z – Zero
V – Overflow
C – Carry
X – Extend

The first four bits are true condition code bits in that they reflect the condition of the result of a processor operation. The X-bit is an operand for multiprecision computations. The carry bit (C) and the multiprecision operand extend bit (X) are separate in the HD68000 to simplify the programming model.

• CONDITION CODE REGISTER NOTATION

In the instruction set details, the description of the effect on the condition codes is given in the following form:

Condition Codes:



Where

N (negative) set if the most significant bit of the result is set. Cleared otherwise.

Z (zero) set if the result equals zero. Cleared otherwise.

V (overflow) set if there was an arithmetic overflow. This implies that the result is not representable in the operand size. Cleared otherwise.

C (carry) set if a carry is generated out of the most significant bit of the operands for an addition. Also set if a borrow is generated in a subtraction. Cleared otherwise.

X (extend) transparent to data movement. When affected, it is set the same as the C-bit.

The notational convention that appears in the representation of the condition code registers is:

* set according to the result of the operation
– not affected by the operation
0 cleared
1 set
U undefined after the operation

• CONDITION CODE COMPUTATION

Most operations take a source operand and a destination operand, compute, and store the result in the destination location. Unary operations take a destination operand, compute, and store the result in the destination location. Table 22 details how each instruction sets the condition codes.

Table 22 Condition Code Computations

Operations	X	N	Z	V	C	Special Definition
ABCD	*	U	?	U	?	C = Decimal Carry $Z = Z \cdot \overline{Rm} \cdot \ldots \cdot \overline{R0}$
ADD, ADDI, ADDQ	*	*	*	?	?	$V = Sm \cdot Dm \cdot \overline{Rm} + \overline{Sm} \cdot \overline{Dm} \cdot Rm$ $C = Sm \cdot Dm + \overline{Rm} \cdot Dm + Sm \cdot \overline{Rm}$
ADDX	*	*	?	?	?	$V = Sm \cdot Dm \cdot \overline{Rm} + \overline{Sm} \cdot \overline{Dm} \cdot Rm$ $C = Sm \cdot Dm + \overline{Rm} \cdot Dm + Sm \cdot \overline{Rm}$ $Z = Z \cdot \overline{Rm} \cdot \ldots \cdot \overline{R0}$
AND, ANDI, EOR, EORI, MOVEQ, MOVE, OR, ORI, CLR, EXT, NOT, TAS, TST	–	*	*	0	0	
CHK	–	*	U	U	U	
SUB, SUBI SUBQ	*	*	*	?	?	$V = \overline{Sm} \cdot Dm \cdot \overline{Rm} + Sm \cdot \overline{Dm} \cdot Rm$ $C = Sm \cdot \overline{Dm} + Rm \cdot \overline{Dm} + Sm \cdot Rm$
SUBX	*	*	?	?	?	$V = \overline{Sm} \cdot Dm \cdot \overline{Rm} + Sm \cdot \overline{Dm} \cdot Rm$ $C = Sm \cdot \overline{Dm} + Rm \cdot \overline{Dm} + Sm \cdot Rm$ $Z = Z \cdot \overline{Rm} \cdot \ldots \cdot \overline{R0}$
CMP, CMPI, CMPM	–	*	*	?	?	$V = \overline{Sm} \cdot Dm \cdot \overline{Rm} + Sm \cdot \overline{Dm} \cdot Rm$ $C = Sm \cdot \overline{Dm} + Rm \cdot \overline{Dm} + Sm \cdot Rm$
DIVS, DIVU	–	*	*	?	0	V = Division Overflow
MULS, MULU	–	*	*	0	0	
SBCD, NBCD	*	U	?	U	?	C = Decimal Borrow $Z = Z \cdot \overline{Rm} \cdot \ldots \cdot \overline{R0}$
NEG NEGX	* *	* *	* ?	? ?	? ?	$V = Dm \cdot Rm$, $C = Dm + Rm$ $V = Dm \cdot Rm$, $C = Dm + Rm$ $Z = Z \cdot \overline{Rm} \cdot \ldots \cdot \overline{R0}$
BTST, BCHG, BSET, BCLR	–	–	?	–	–	$Z = \overline{Dn}$
ASL	*	*	*	?	?	$V = Dm \cdot (\overline{D_{m-1}} + \ldots + \overline{D_{m-r}}) + \overline{Dm} \cdot (D_{m-1} + \ldots + D_{m-r})$ $C = D_{m-r+1}$
ASL (r = 0)	–	*	*	0	0	
LSL, ROXL	*	*	*	0	?	$C = D_{m-r+1}$
LSR (r = 0)	–	*	*	0	0	
ROXL (r = 0)	–	*	*	0	?	C = X
ROL	–	*	*	0	?	$C = D_{m-r+1}$
ROL (r = 0)	–	*	*	0	0	
ASR, LSR, ROXR	*	*	*	0	?	$C = D_{r-1}$
ASR, LSR (r = 0)	–	*	*	0	0	
ROXR (r = 0)	–	*	*	0	?	C = X
ROR	–	*	*	0	?	$C = D_{r-1}$
ROR (r = 0)	–	*	*	0	0	

– Not affected
U Undefined
? Other— see Special Definition

* General Case
X = C
N = Rm
$Z = \overline{Rm} \cdot \ldots \cdot \overline{R0}$

Sm – Source operand most significant bit
Dm – Destination operand most significant bit
Rm – Result bit most significant bit
n – bit number
r – shift amount

• CONDITIONAL TESTS

Table 23 lists the condition names, encodings, and tests for the conditional branch and set instructions. The test associated with each condition is a logical formula based on the current state of the condition codes. If this formula evaluates to 1, the condition succeeds, or is true. If the formula evaluates to 0, the condition is unsuccessful, or false. For example, the T condition always succeeds, while the EQ condition succeeds only if the Z bit is currently set in the condition codes.

Table 23 Conditional Tests

Mnemonic	Condition	Encoding	Test
T	true	0000	1
F	false	0001	0
HI	high	0010	$\overline{C} \cdot \overline{Z}$
LS	low or same	0011	$C + Z$
CC	carry clear	0100	$\overline{C}$
CS	carry set	0101	C
NE	not equal	0110	$\overline{Z}$
EQ	equal	0111	Z
VC	overflow clear	1000	$\overline{V}$
VS	overflow set	1001	V
PL	plus	1010	$\overline{N}$
MI	minus	1011	N
GE	greater or equal	1100	$N \cdot V + \overline{N} \cdot \overline{V}$
LT	less than	1101	$N \cdot \overline{V} + \overline{N} \cdot V$
GT	greater than	1110	$N \cdot V \cdot \overline{Z} + \overline{N} \cdot \overline{V} \cdot \overline{Z}$
LE	less or equal	1111	$Z + N \cdot \overline{V} + \overline{N} \cdot V$

• INSTRUCTION SET

The following paragraphs provide information about the addressing categories and instruction set of the HD68000.

• ADDRESSING CATEGORIES

Effective address modes may be categorized by the ways in which they may used. The following classifications will be used in the instruction definitions.

Data — If an effective address mode may be used to refer to data operands, it is considered a data addressing effective address mode.

Memory — If an effective address mode may be used to refer to memory operands, it is considered a memory addressing effective address mode.

Alterable — If an effective address mode may be used to refer to alterable (writeable) operands, it is considered an alterable addressing effective address mode.

Control — If an effective address mode may be used to refer to memory operands without an associated size, it is considered a control addressing effective address mode.

Table 24 shows the various categories to which each of the effective address modes belong. Table 25 is the instruction set summary.

The status register addressing mode is not permitted unless it is explicitly mentioned as a legal addressing mode.

These categories may be combined so that additional, more restrictive, classifications may be defined. For example, the instruction descriptions use such classifications as alterable memory or data alterable. The former refers to those addressing modes which are both alterable and memory addresses, and the latter refers to addressing modes which are both data and alterable.

• INSTRUCTION PRE-FETCH

The HD68000 uses a 2-word tightly-coupled instruction prefetch mechanism to enhance performance. This mechanism is described in terms of the microcode operations involved. If the execution of an instruction is defined to begin when the microroutine for that instruction is entered, some features of the prefetch mechanism can be described.

1) When execution of an instruction begins, the operation word and the word following have already been fetched. The operation word is in the instruction decoder.
2) In the case of multi-word instructions, as each additional word of the instruction is used internally, a fetch is made to the instruction stream to replace it.
3) The last fetch from the instruction stream is made when the operation word is discarded and decoding is started on the next instruction.
4) If the instruction is a single-word instruction causing a branch, the second word is not used. But because this word is fetched by the preceding instruction, it is impossible to avoid this superfluous fetch. In the case of an interrupt or trace exception, both words are not used.
5) The program counter usually points to the last word fetched from the instruction stream.

Table 24 Effective Addressing Mode Categories

Effective Address Modes	Mode	Register	Data	Addressing Categories		
				Memory	Control	Alterable
Dn	000	register number	X	–	–	X
An	001	register number	–	–	–	X
An@	010	register number	X	X	X	X
An@+	011	register number	X	X	–	X
An@–	100	register number	X	X	–	X
An@(d)	101	register number	X	X	X	X
An@(d, ix)	110	register number	X	X	X	X
xxx.W	111	000	X	X	X	X
xxx.L	111	001	X	X	X	X
PC@(d)	111	010	X	X	X	–
PC@(d, ix)	111	011	X	X	X	–
#xxx	111	100	X	X	–	–

The following example illustrates many of the features of instruction prefetch. The contents of memory are assumed to be as illustrated in Figure 53.

```
            ORG     0                       DEFINE RESTART VECTOR
            DC.L    INISSP                  INITIAL SYSTEM STACK POINTER
            DC.L    RESTART                 RESTART SYSTEM ENTRY POINT
            ORG     INTVECTOR               DEFINE AN INTERRUPT VECTOR
            DC.L    INTHANDLER              HANDLER ADDRESS FOR THIS VECTOR
            ORG                             SYSTEM RESTART CODE
RESTART:
            NOP                             NO OPERATION EXAMPLE
            BRA.S   LABEL                   SHORT BRANCH
            ADD.W   D0, D1                  ADD REGISTER TO REGISTER
LABEL:
            SUB.W   DISP(A0), A1            SUBTRACT REGISTER INDIRECT WITH OFFSET
            CMP.W   D2, D3                  COMPARE REGISTER TO REGISTER
            SGE.B   D7                      Scc TO REGISTER
            . . .
            . . .
INTHANDLER:
            MOVE.W  LONGADR1, LONGADR2      MOVE WORD FROM AND TO LONG ADDRESS
            NOP                             NO OPERATION
            SWAP.W                          REGISTER SWAP
```

Figure 53 Instruction Prefetch Example, Memory Contents

The sequence we shall illustrate consists of the power-up reset, the execution of NOP, BRA, SUB, the taking of an interrupt, and the execution of the MOVE.W xxx. L to yyy.L. The order of operations described within each microroutine is not exact, but is intended for illustrative purpose only.

Microroutine	Operation	Location	Operand
Reset	Read	0	SSP High
	Read	2	SSP Low
	Read	4	PC High
	Read	6	PC Low
	Read	(PC)	NOP
	Read	+(PC)	BRA
	<begin NOP>		
NOP	Read	+(PC)	ADD
	<begin BRA>		
BRA	PC=PC+d		
	Read	(PC)	SUB
	Read	+(PC)	DISP
	<begin SUB>		
SUB	Read	+(PC)	CMP
	Read	DISP(A0)	<src>
	Read	+(PC)	SGE
	<begin CMP>	<take INT>	
INTERRUPT	Write	-(SSP)	PC Low
	Read	<INT ACK>	Vector #
	Write	-(SSP)	SR
	Write	-(SSP)	PC High
	Read	(VR)	PC High
	Read	+(VR)	PC Low
	Read	(PC)	MOVE
	Read	+(PC)	xxx High
	<begin MOVE>		
MOVE	Read	+(PC)	xxx Low
	Read	+(PC)	yyy High
	Read	xxx	<src>
	Read	+(PC)	yyy Low
	Write	yyy	<dest>
	Read	+(PC)	NOP
	Read	+(PC)	SWAP
	<begin NOP>		

Figure 54 Instruction Prefetch Example

• DATA PREFETCH

Normally the HD68000 prefetches only instructions and not data. However, when the MOVEM instruction is used to move data from memory to registers, the data stream is prefetched in order to optimize performance. As a result, the processor reads one extra word beyond the higher end of the source area. For example, the instruction sequence in Figure 55 will operate as shown in Figure 56.

	...		MOVE TWO LONGWORDS
	MOVEM.L	A, D0/D1	INTO REGISTERS
	...		
A	DC.W	1	WORD 1
B	DC.W	2	WORD 2
C	DC.W	3	WORD 3
D	DC.W	4	WORD 4
E	DC.W	5	WORD 5
F	DC.W	6	WORD 6

Figure 55 MOVEM Example, Memory Contents

Assume Effective Address Evaluation is Already Done

Microroutine	Operation	Location	Other Operations
	...		
MOVEM	Read	A	
			Prepare to Fill D0
	Read	B	A → D0H
	Read	C	B → D0L
			Prepare to Fill D1
	Read	D	C → D1H
	Read	E	D → D1L
			Detect Register List Complete

Figure 56 MOVEM Example, Operation Sequence

Table 25 Instruction Set

Mnemonic Operation	Size	Addr. Mode	Dn #	Dn ~	An #	An ~	(An) #	(An) ~	(An)+ #	(An)+ ~	−(An) #	−(An) ~	d(An) #	d(An) ~	d(An,Xi) #	d(An,Xi) ~	Abs.W #	Abs.W ~	Abs.L #	Abs.L ~	d(PC) #	d(PC) ~	d(PC,Xi) #	d(PC,Xi) ~	#=Immed d=SR/CC #	#=Immed d=SR/CC ~	Opcode Bit Pattern 1111 11 0432 1098 7654 3210	Boolean	Condition Codes XNZVC
ABCD	B	s:Dn d	2	6																						1100 RRR1 0000 0rrr	d10 ← s10 + X + d	* U * U *
Add Decs		s −(An) d									2	18														1100 RRR1 0000 1rrr		
ADD	B/W	s:Dn d:			ADDA		2	12	2	12	2	14	4	16	4	18	4	16	6	20						1101 DDD1 SSEE EEEE	d + Dn → d	* * * * *
Add		d:Dn s:	2	4	2*	4	2	8	2	8	2	10	4	12	4	14	4	12	6	16	4	12	4	14	4	8	1101 DDD0 SSee eeee	Dn + s → Dn	
Binary	L	s:Dn d			ADDA		2	20	2	20	2	22	4	24	4	26	4	24	6	28						1101 DDD1 10EE EEEE	d + Dn → d	
		d:Dn s:	2	8	2	8	2	14	2	14	2	16	4	18	4	20	4	18	6	22	4	18	4	20	6	14	1101 DDD0 10ee eeee	Dn + s → Dn	
ADDA	W	s:An d	2	8	2	8	2	12	2	12	2	14	4	16	4	18	4	16	6	20	4	16	4	18	4	12	1101 AAA0 11ee eeee	An + s → An	- - - - -
Add Address	L	d:An s	2	8	2	8	2	14	2	14	2	16	4	18	4	20	4	18	6	22	4	18	4	20	6	14	1101 AAA1 11ee eeee		
ADDI	B/W	s:Imm d:	4	8	ADDA		4	16	4	16	4	18	6	20	6	22	6	20	8	24						0000 0110 SSEE EEEE	d + #s → d	* * * * *
Add Immed	L	s:Imm d:	6	16	ADDA		6	28	6	28	6	30	8	32	8	34	8	32	10	36								
ADDQ	B W	s:Imm3 d	2	4			2	12	2	12	2	14	4	16	4	18	4	16	6	20						0101 QQQ0 SSEE EEEE	s + #s → d	* * * * *
Add Quick		s:Imm3 d	2	8	2	8	2	8	2	8	2	8	4	24	4	26	4	24	6	28								
ADDX	B/W	s:Dn d:	2	4																						1101 RRR1 SS00 0rrr	d + s + X → d	* * * * *
Add Multi-precision		s: −(An) d:									2	18														1101 RRR1 SS00 1rrr		
	L	s:Dn d:	2	8																						1101 RRR1 0000 0rrr		
		s: −(An) d:									2	30														1101 RRR1 0000 1rrr		
AND	B W	s:Dn d					2	12	2	12	2	14	4	16	4	18	4	16	6	20						1100 DDD1 SSEE EEEE	d < and > Dn → d	- * * 0 0
Logical And		d:Dn s	2	4			2	8	2	8	2	10	4	12	4	14	4	12	6	16	4	12	4	14	4	8	1100 DDD0 SSee eeee	Dn < and > s → Dn	
	L	s:Dn d					2	20	2	20	2	22	4	24	4	26	4	24	6	28						1100 DDD1 10EE EEEE	d < and > Dn → d	
		d:Dn s	2	8			2	14	2	14	2	16	4	18	4	20	4	18	6	22	4	18	4	20	6	14	1100 DDD0 10ee eeee	Dn < and > s → Dn	
ANDI	B/W	s:Imm d:	4	8			4	16	4	16	4	18	6	20	6	22	6	20	8	24					4	20	0000 0010 SSEE EEEE	d < and > #s → d	- * * 0 0
And Immed	L	s:Imm d:	6	16			6	28	6	28	6	30	8	32	8	34	8	32	10	36								
ASL, ASR	B/W	count:Dn d	2	6+2n																						1110 rrrf SS10 0DDD	C ← X ← 0 (Left); 0 → X → C (Right)	* * * * *
Arithmetic		count:#1–8 d	2	6+2n																						1110 QQQf SS00 0DDD		
Shift	L	count:Dn d	2	8+2n																						1110 rrrf 0110 0DDD		
		count:#1–8 d	2	8+2n																						1110 QQQf 1000 0DDD		
Memory	W	count:1 ea					2*	12	2*	12	2*	14	4*	16	4*	18	4*	16	6*	20						1110 000f 11EE EEEE		
BCHG	B	bit #:Dn d:					2	12	2	12	2	14	4	16	4	18	4	16	6	20						0000 rrr1 01EE EEEE	~(bit) # of d → Z, ~(bit) # of d → (bit) # of d	- * * - -
Test and		bit #:Imm d:					4	16	4	16	4	18	6	20	6	22	6	20	8	24						0000 1000 01FE EEEE		
Change	L	bit #:Dn d:	2	<8																						0000 rrr1 01EE EEEE		
		bit #:Imm d:	4	<12																						0000 1000 01EE EEEE		
BCLR	B	bit #:Dn d:					2	12	2	12	2	14	4	16	4	18	4	16	6	20						0000 rrr1 10EE EEEE	~(bit) # of d → Z, 0 → (bit) # of d	- - * - -
Test and		bit #:Imm d:					4	16	4	16	4	18	6	20	6	22	6	20	8	24						0000 1000 10EE EEEE		
Clear		bit #:Dn d:	2	<10																						0000 rrr1 10EE EEEE		
		bit #:Imm d	4	<14																						0000 1000 10EE EEEE		
BSET	B	bit #:Dn d:					2	12	2	12	2	14	4	16	4	18	4	16	6	20						0000 rrr1 11EE EEEE	~(bit) # of d → Z, 1 → (bit) # of d	- - * - -
Test and		bit #:Imm d:					4	16	4	16	4	18	6	20	6	22	6	20	8	24						0000 1000 11EE EEEE		
Set	L	bit #:Dn d:	2	<8																						0000 rrr1 11EE EEEE		
		bit #:Imm d:	4	<12																						0000 1000 11EE EEEE		
BTST	B	bit #: Dn d					2	8	2	8	2	10	4	12	4	14	4	12	6	16	4	12	4	14	4	10	0000 rrr1 00EE EEEE	~(bit) # of d → Z	- - * - -
Bit Test		bit #:Imm d:					4	12	4	12	4	14	6	16	6	18	6	16	8	20	6	16	6	18			0000 1000 00EE EEEE		
	L	bit #:Dn d	2	6																						0000 rrr1 00EE EEEE		
		bit #:Imm d	4	10																						0000 1000 00EE EEEE		
CHK	W			<40	← trap →			<44		<44		<46		<48		<50		<48		<52		<48		<50		<44	0100 DDD1 10ee eeee	If Dn < 0, or Dn > (bound) then trap	- * U U *
Check Reg ister Against Bounds		d:Dn s: (bound)	2	10	← no → trap		2	14	2	14	2	16	4	18	4	20	4	18	6	22	4	18	4	20	4	14			
CLR	B W	d	2	4			2	12	2	12	2	14	4	16	4	18	4	16	6	20						0100 0010 SSEE EEEE	d ← #0L	- 0 1 0 0
Clear Operand		d	2	6			2	8	2	8	2	10	4	12	4	14	4	12	6	16	4	12	4	14	4	8		0 → d	
CMP	B/W	d:Dn s	2	4	2*	4	2	8	2	8	2	10	4	12	4	14	4	12	6	16	4	12	4	14	4	8	1011 DDD0 SSee eeee	Dn − s	- * * * *
Compare	L	d:Dn s	2	6	2	6	2	14	2	14	2	16	4	18	4	20	4	18	6	22	4	18	4	20	6	14			
Binary																													
CMPA	W	d:An s	2	6	2	6	2	10	2	10	2	12	4	14	4	16	4	12	6	18	4	14	4	16	4	10	1011 AAA0 11ee eeee	An − s	- * * * *
Compare	L	d:An s	2	6	2	6	2	14	2	14	2	16	4	18	4	20	4	18	6	22	4	18	4	20	6	14	1011 AAA1 11ee eeee		
Address																													
CMPI	B/W	s:Imm d:	4	8	CMPA		4	12	4	12	4	14	6	16	6	18	6	16	8	20						0000 1100 SSEE EEEE	d − #s	- * * * *
Compare Imm	L	s:Imm d:	6	14	CMPA		6	20	6	20	6	22	8	24	8	26	8	24	10	28								
CMPM	B W	s: An + d:						12																		1011 RRR1 SS00 1rrr	d − s	- * * * *
Compare Memory		s: (An) + d						20																				
DIVS	W	d:Dn s:	2	<158			2	<162	2	<162	2	<164	4	<166	4	<168	4	<166	6	<170	4	<166	4	<168	4	<162	1000 DDD1 11ee eeee	Dn32/s16 → Dn(rq)	- * * * *
Divide Signed																													
DIVU	W	d:Dn s	2	<140			2	<144	2	<144	2	<146	4	<148	4	<150	4	<148	6	<152	4	<148	4	<150	4	<144	1000 DDD0 11ee eeee	Dn32/s16 → Dn(rq)	- * * * 0
Divide Unsigned																													
EOR	B/W	s:Dn d:	2	4			2	12	2	12	2	14	4	16	4	18	4	16	6	20						1011 rrr1 SSEE EEEE	d ⊕ Dn → d	- * * 0 0
Exclusive OR Logical	L	s:Dn d:	2	8			2	20	2	20	2	22	4	24	4	26	4	24	6	28								
EORI	B W	s:Imm d	4	8			4	16	4	16	4	18	6	20	6	22	6	20	8	24					4	20	0000 1010 SSEE EEEE	d ⊕ #s → d	- * * 0 0
Exclusive OR Immediate	L	s:Imm d	6	16			6	28	6	28	6	30	8	32	8	34	8	32	10	36								
EXG	L	s:Dn d:	2	6																						1100 DDD1 0100 0DDD	s ↔ d	- - - - -
Exchange Registers		s:An d	2	6	2	6																				1100 AAA1 0100 1AAA		
																										1100 DDD1 1000 1AAA		
EXT	W	d		4																						0100 1000 1000 0DDD	bit 7 → bit 8 ... 15	
Sign Extend	L	d		4																						0100 1000 1100 0DDD	bit 15 → bit 16 ... 31	- * * 0 0
LEA	L	d:An s:					2	4					4	8	4	12	4	8	6	12	4	8	4	12			0100 AAA1 11ee eeee	s → An	- - - - -
Load Effect ive Address																													
LINK		disp:Imm s				4	16																			0100 1110 010r 0AAA	An ← SP; SP ← An; SP + disp → SP	- - - - -
Link and Allocate																													

Note: Refer to Condition Code Computations as for condition Code
@ Word only
< Maximum value
Number of Program Bytes
← Number of Clock Periods

Opcode Bit Pattern Key

A Address Register #
C Test Condition
D Data Register #
e Source Effective Address
E Destination Effective Address
I Direction 0 - Right 1 - Left
M Destination EA Mode
P Displacement
Q Quick Immediate Data
r Source Register
R Destination Register
S Size 00 - Byte; 01 - Word; 10 - Long Word; 11 - Another Operation
V Vector #

* in the MOVE instruction
0 - Byte
10 - Long Word
11 - Word

(to be continued)

Mnemonic Operation	Size	Addr. Mode	Dn		An		(An)		(An)+		-(An)		d(An)		d(An,Xi)		Abs.W		Abs.L		d(PC)		d(PC,Xi)		#=immed d=SR/CC		Opcode Bit Pattern 1111 11 5432 1098 7654 3210	Sentence	Condition Codes XNZVC
			#	~	#	~	#	~	#	~	#	~	#	~	#	~	#	~	#	~	#	~	#	~	#	~		
LSL, LSR Logical Shift	B/W	count Dn d	2	6+2n																							1110 rrrf SS10 1DDD	C ←0 X ← Left 0 →C X Right	●●●0●
	L	count #1~8 d	2	6+2n																							1110 QQQf SS00 1DDD		
		count Dn d	2	8+2n																							1110 rrrf 1010 1DDD		
		count #1~8 d	2	8+2n																							1110 QQQf 1000 1DDD		
Memory	W	count 1 d					2*	12	2*	12	2*	14	4*	16	4*	18	4*	16	6*	20							1110 001 1 11EE EEEE		
MOVE Move Data	B/W	s,Dn d	2	4	MOVEA		2	8	2	8	2	8	4	12	4	14	4	12	6	16							00SS RRRM MMee eeee	s→d	- ●● 0 0
		s,(An) d	2	4	MOVEA		2	8	2	8	2	8	4	12	4	14	4	12	6	16									
		s,d(An) d	2	8	MOVEA		2	12	2	12	2	12	4	16	4	18	4	16	6	20									
		s,d(An)+ d	2	8	MOVEA		2	12	2	12	2	12	4	16	4	18	4	16	6	20									
		s,-(An) d	2	10	MOVEA		2	14	2	14	2	14	4	18	4	20	4	18	6	22									
		s,d(An) d	4	12	MOVEA		4	16	4	16	4	16	6	20	6	22	6	20	8	24									
		s,d(An,Xi) d	4	14	MOVEA		4	18	4	18	4	18	6	22	6	24	6	22	8	26									
		s,Abs.W d	4	12	MOVEA		4	16	4	16	4	16	6	20	6	22	6	20	8	24									
		s,Abs.L d	6	16	MOVEA		6	20	6	20	6	20	8	24	8	26	8	24	10	28									
		s,d(PC) d	4	12	MOVEA		4	16	4	16	4	16	6	20	6	22	6	20	8	24									
		s,d(PC,Xi) d	4	14	MOVEA		4	18	4	18	4	18	6	22	6	24	6	22	8	26									
		s,#xxx d	4	8	MOVEA		4	12	4	12	4	12	6	16	6	18	6	16	8	20									
	L	s,Dn d					2	12	2	12	2	12	4	16	4	18	4	16	6	20									
		s,An d					2	12	2	12	2	12	4	16	4	18	4	16	6	20									
		s,(An) d					2	20	2	20	2	20	4	24	4	26	4	24	6	28									
		s,(An)+ d					2	20	2	20	2	20	4	24	4	26	4	24	6	28									
		s,-(An) d					2	22	2	22	2	22	4	26	4	28	4	26	6	30									
		s,d(An) d					4	24	4	24	4	24	6	28	6	30	6	28	8	32									
		s,d(An,Xi) d					4	26	4	26	4	26	6	30	6	32	6	30	8	34									
		s,Abs.W d					4	24	4	24	4	24	6	28	6	30	6	28	8	32									
		s,Abs.L d					6	28	6	28	6	28	8	32	8	34	8	32	10	36									
		s,d(PC) d					4	24	4	24	4	24	6	28	6	30	6	28	8	32									
		s,d(PC,Xi) d					4	26	4	26	4	26	6	30	6	32	6	30	8	34									
		s,#nnn d					6	20	6	20	6	20	8	24	8	26	8	24	10	28									
MOVE Move to Con ditions Codes	W	d,CCR s	2	12																							0100 0100 11ee eeee	s →CCR	●●●●●
MOVE Move to/from Status Reg	W	d,SR s	2	12																							0100 0110 11ee eeee	s →SR	●●●●●
		s,SR d	2	6																							0100 0000 11EE EEEE	d →MOVE SR →d	- - - - -
MOVE Move to/from User SP(A7)	L	s,USP d			2	4																					0100 1110 0110 1AAA	USP →An	- - - - -
		d,USP s			2	4																					0100 1110 0110 0AAA	An →USP	
MOVEA Move Address	W	d,An s	2	4	2	4	2	8	2	8	2	10	4	12	4	14	4	12	6	16	4	12	4	14	4	8	0011 AAA0 01ee eeee	s →An	- - - - -
		d,An s	2	4	4	4	2	8	2	8	2	10	4	12	4	14	4	12	6	16	4	12	4	14	4	8	0010 AAA0 01ee eeee		
MOVEM Move Multiple Registers	W	s,list d→						4 8+4n			4	8+4n	6	12+4n	6	14+4n	6	12+4n	8	16+4n							0100 1000 10EE EEEE	Rn →d	- - - - -
																											a7 a0 d7~d0		
		d list s→						4 12+4n	4	12+4n			6	16+4n	6	18+4n	6	16+4n	8	20+4n	6	16+4n	6	18+4n			0100 1100 10ee eeee	s →Rn**	
																											a7 a0 d7~d0		
		L s,Rn d						4 8+8n		4 8+8n			6	12+6n	6	14+8n	6	12+8n	8	16+8n							0100 1000 11EE EEEE	Rn →d	
																											a7 a0 d7~d0		
		d→list s						4 12+8n	4	12+8n			6	16+8n	6	18+8n	6	16+8n	8	20+8n	6	16+8n	6	18+8n			0100 1100 11ee eeee	s →Rn**	
																											a7 a0 d7~d0		
MOVEP Move Periphral	W	s,Dn d											4	16													0000 DDD1 1000 1AAA	Dn →d by bytes	- - - - -
		s,d(An) d	4	16																							0000 DDD1 0000 1AAA	s →Dn by bytes	
	L	s,Dn d											4	24													0000 DDD1 1100 1AAA	Dn →d by bytes	
		s,d(An) d	4	24																							0000 DDD1 0100 1AAA	s →Dn by bytes	
MOVEQ Move Quick	L	s,Dn d	2	4																							0111 DDD0 0QQQ QQQQ	#≤ →Dn	- ●●0 0
MULS Multiply Signed	W	d,Dn s	2	~70			2	~74	2	~74	2	~76	4	~78	4	~80	4	~78	6	~82	4	~78	4	~80	4	~74	1100 DDD1 11ee eeee	Dn ×s →Dn	- ●●0 0
MULU Multiply Unsigned	W	d,Dn s	2	<70			2	<74	2	<74	2	<76	4	<78	4	<80	4	<78	6	<82	4	<78	4	<80	4	<74	1100 DDD0 11ee eeee	Dn ×s →Dn	- ●●0 0
NBCD Negate Decimal	B	d 0	2	6			2		2		2		4		4		4		6								0100 1000 00EE EEEE	0 −d10 −x →d	●. ●. ●. ●
NEG Negate Binary	B/W	d 0	2	4			2	12	2	12	2	14	4	16	4	18	4	16	6	20							0100 0100 SSEE EEEE	0 −d →d	●●●●●
	L	d 0	2	6			2	20	2	20	2	22	4	24	4	26	4	24	6	28									
NEGX Negate Multi precision	B/W	d 0	2	4			2		2		2		4		4		4		6								0100 0000 SSEE EEEE	0 −d −x →d	●●●●●
	L	d 0	2	6			2		2		2		4		4		4		6										
NOT Logical Complement	B/W	d 0	2	4			2	12	2	12	2	14	4	16	4	18	4	16	6	20							0100 0110 SSEE EEEE	d <or> Dn →d	- ●● 0 0
	L	d 0	2	6			2	20	2	20	2	22	4	24	4	26	4	24	6	28									
OR Inclusive OR Logical	B/W	s,Dn d					2		2		2		4		4		4		6								1000 DDD1 SSEE EEEE	Dn <or> s →Dn	●● 0 0
		d,Dn s	2	4			2	8	2	8	2	10	4	12	4	14	4	12	6	16	4	12	4	14	4	8	1000 DDD1 10EE EEEE	d <or> Dn →d	
		s,Dn d	2	4			2		2		2		4		4		4		6								1000 DDD0 11ee eeee	Dn <or> s →Dn	
ORI Inclusive OR Immediate	B/W	s,Dn d	4	8			4	16	4	16	4	18	6	20	6	22	6	20	8	24							0000 0000 SSEE EEEE	#<or> d →d	- ●●0 0
OR Immediate	L	s,Dn d	6	16			6	28	6	28	6	30	8	32	8	34	8	32	10	36									
PEA Push Effect ive Address		s					2	12					4	16	4	18	4	16	6	20							0100 1000 01ee eeee	s →-(SP)	- - - - -
RESET, RTE, RTR, RTS, STOP	B/W	count Dn d	2	6+2n																							1110 rrrf SS11 1DDD	C Right n	- ●●●●
		count #1~8 d	2	6+2n																							1110 QQQf SS01 1DDD		
	L	count Dn d	2	8+2n																							1110 rrrf 1011 1DDD	C Left n	
		count #1~8 d	2	8+2n																							1110 QQQf 1001 1DDD		
Illegal	W	count 1 d																									1110 0111 11EE EEEE		

Note Refer to Condition Code Computations as for condition Code

- o Word only
- • Maximum value
- # Number of Program Bytes
  Number of Clock Periods
- ●● The MPU goes through an extra null read cycle after a multiple read a done (The last EA+2)

Opcode Bit Pattern Key

A Address Register S
C Test Condition
D Data Register D
e Source Effective Address
E Destination Effective Address
f Direction 0 Right 1 Left
M Destination EA Mode
P Displacement
Q Quick Immediate Data
r Source Register
R Destination Register
S Size 00 - Byte
01 - Word
10 - Long Word
11 - Another Operation
V Vector #

(In the MOVE instruction)
01 Byte
10 - Long Word
11 - Word

(to be continued)

Mnemonic Operation	Size	Addr. Mode	Dn		An		(An)		(An)+		-(An)		d(An)		d(An,Xi)		Abs.W		Abs.L		d(PC)		d(PC,Xi)		#=Immed d=SR/CC		Opcode Bit Pattern 1111 11 5432 1098 7654 3210	Bestem	Condition Codes XNZVC
			#	~	#	~	#	~	#	~	#	~	#	~	#	~	#	~	#	~	#	~	#	~	#	~			
ROXR,ROXL Rotate through X	B W	count Dn d	2	6+2n																							1110 rrrf SS11 0DDD		●●●○●
	L	count #1~8 d	2	6+2n																							1110 QQQf SS01 0DDD		
		count Dn d	2	8+2n																							1110 rrrf 1011 0DDD		
		count #1~8 d	2	8+2n																							1110 QQQf 1001 0DDD		
Memory	W	count 1 d					2*	12	2*	12	2*	14	4*	16	4*	18	4*	16	6*	20							1110 0101 11EE EEEE		
SBCD	B	s-Dn d=	2	6																							1000 RRR1 0000 0rrr		
Subtract		s- (An) d=									2	18															1000 RRR1 0000 1rrr	d10 s10 X →d	●U●●●
digits																													
Scc Set Conditionally	B	cc d	2	6-4			2	12	2	12	2	14	4	16	4	18	4	16	6	20							0101 CCCC 11EE EEEE	d →MPU If cc true then 1s →d Else 0s →d	- - - - -
SUB	B W	s-Dn d=			SUBA		2	12	2	12	2	14	4	16	4	18	4	16	6	20							1001 DDD1 SSEE EEEE	d-Dn →d	●●●●●
Subtract		s-Dn s-	2	4	2*	4	2	8	2	8	2	10	4	12	4	14	4	12	6	16	4	12	4	14	4	8	1001 DDD0 SSee eeee	Dn s →Dn	
Binary	L	s-Dn d=			SUBA		2	20	2	20	2	22	4	24	4	26	4	24	6	28							1001 DDD1 11EE EEEE	d-Dn →d	
		d-Dn s=	2	8	2	8	2	14	2	14	2	16	4	18	4	20	4	18	6	22	4	18	4	20	6	14	1001 DDD0 10ee eeee	Dn s →Dn	
SUBA	W	d-An s	2	8	2	8	2	12	2	12	2	14	4	16	4	18	4	16	6	20	4	16	4	18	4	12	1001 AAA0 11ee eeee	An s →An	
Subtract Address	L	d An s	2	8	2	8	2	14	2	14	2	16	4	18	4	20	4	18	6	22	4	18	4	20	6	14	1001 AAA1 11ee eeee		- - - - -
SUBI	B W	s-imm d=	4	8	SUBA		4	16	4	16	4	18	6	20	6	22	6	20	8	24							0000 0100 SSEE EEEE	d- # →d	●●●●●
Subtract	L	s-imm d=	6	16	SUBA		6	28	6	28	6	30	8	32	8	34	8	32	10	36									
Immediate																													
SUBQ	B W	s Imm3 d	2	4	2*	4	2	12	2	12	2	14	4	16	4	18	4	16	6	20							0101 QQQ1 SSEE EEEE	d # →d	●●●●●
Subtract Quick	L	s Imm3 d	2	8	2	8	2	16	2	16	2	22	4	24	4	26	4	24	6	28									
SUBX	B W	s-Dn d=	2	4																							1001 RRR1 SS00 0rrr	d s X →d	●●●●●
Subtract		s- (An) d=									2	18															1001 RRR1 SS00 1rrr		
Multiprecession	L	s-Dn d=	2	8																							1001 RRR1 1000 0rrr		
		s- (An) d=									2	30															1001 RRR1 1000 1rrr		
SWAP Swap Registers Halves	W	d	2	4																							0100 1000 0100 0DDD	Dn(31:16) ⟷ Dn(15:0)	- ●●0 0
TAS Test and Set Operand	B	d=	2	4			2	14	2	14	2	16	4	18	4	20	4	18	6	22							0100 1010 11EE EEEE	test d →cc 1 →bit 7 of d	- ●●0 0
TST Test	B W L	d	2	4			2	8	2	8	2	10	4	12	4	14	4	12	6	16							0100 1010 SSEE EEFF	test d →cc	- ●●0 0
		d	2	4			2	12	2	12	2	4	4	16	4	18	4	16	6	20									
UNLK Unlink					2	12																					0100 1110 0101 1AAA	An →SP (SP)+ →An	- - - - -

Bcc Branch Conditionally	B	disp																			bra taken	2	10	0110 CCCC	PPPP PPPP	if cc true PC+disp →PC	- - - - -
	W	disp																			bra not taken	2	8				
																					bra taken	4	10				
																					bra not taken	4	14				
BRA Branch Always	B	disp																				2	10	0110 0000	PPPP PPPP	PC+disp →PC	- - - - -
	W	disp																				4	10				
BSR Branch to Subroutine	B	disp																				2	20	0110 0001	PPPP PPPP	PC * (SP) PC+disp →PC	- - - - -
	W	disp																				4	20				
DBcc	W	disp-label																						0101 CCCC 1100 1DDD		If cc false Dn 1 →Dn & d Dn= LPC+disp →PC Else NOP	- - - - -
Decrement Counter & Branch Until Condition True or Count =1		counter-	4	10 12 14																									

					cc		Counter		Branch																				
					false true false		n 1 n 1 expired		yes no no																				
JMP Jump to		d									2	8															0100 1110 11 EE EEEE	d →PC	- - - - -
JSR Jump to Subroutine		d=									2	16															0100 1110 10EE EEEE	PC → (SP) d →PC	- - - - -
NOP No Operation			2	4																							0100 1110 0111 0001	none	- - - - -
RESET Reset External Devices			2	132																							0100 1110 0111 0000	assert RESET pin	- - - - -
RTE Return from Exception			2	20																							0100 1110 011. 0011	(SP) →SR (SP) →PC	●●●●●
RTR Return from Subroutine Restore CC			2	20																							0100 1110 0111 0111	(SP) →CC (SP) →PC	●●●●●
RTS Return from Subroutine			2	16																							0100 1110 0111 0101	(SP) →PC	- - - - -
STOP Load SR Stop																						4	4	0100 1110 0111 0010		#→SR Wait for Interrupt	●●●●●
TRAP Trap			2	34																				0100 1110 0100 VVVV		PC → (SSP) SR → (SSP) Vector → PC	- - - - -
TRAPV Trap if Overflow Set			2	34 4	Trap taken Trap not taken																			0100 1110 0111 0110		If V then PC → (SSP) SR → (SSP) (TRAPV vector) →PC else NOP	- - - - -

Note Refer to Condition Code Computations as for condition Code
● Word only
< Maximum value
B Number of Program Bytes
~ Number of Clock Periods

Opcode Bit Pattern Key

A Address Register S
C Test Condition
D Data Register S
e Source Effective Address
E Destination Effective Address
f Direction 0 Right, 1 Left
M Destination EA Mode
P Displacement
Q Quick Immediate Data
r Source Register
R Destination Register
S Size 00 Byte
01 Word
10 Long Word
11 Another Operation
(In the MOVE instruction)
01 Byte
10 Long Word
11 Word
V Vector #

■ INSTRUCTION FORMAT SUMMARY

This provides a summary of the first word in each instruction of the instruction set. Table 26 is an operation code (op-code) map which illustrates how bits 15 through 12 are used to specify the operations. The remaining paragraph groups the instructions according to the op-code map.

where, Size; Byte = 00 Sz; Word = 0
Word = 01 Long Word = 1
Long Word = 10

Table 26 Operation Code Map

Bits 15 thru 12	Operation
0000	Bit Manipulation/MOVEP/Immediate
0001	Move Byte
0010	Move Long
0011	Move Word
0100	Miscellaneous
0101	ADDQ/SUBQ/S_{cc}/DB_{cc}
0110	B_{cc}
0111	MOVEQ
1000	OR/DIV/SBCD
1001	SUB/SUBX
1010	(Unassigned)
1011	CMP/EOR
1100	AND/MUL/ABCD/EXG
1101	ADD/ADDX
1110	Shift/Rotate
1111	(Unassigned)

(1) BIT MANIPULATION, MOVE PERIPHERAL, IMMEDIATE INSTRUCTIONS

Dynamic Bit

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	Register			1	Type		Effective Address					

Static Bit

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	1	0	0	0	Type		Effective Address					

Bit Type Codes: TST = 00, CHG = 01, CLR = 10, SET = 11

MOVEP

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	Register			Op-Mode			0	0	1	Register		

Op-Mode, Word to Reg = 100, Long to Reg = 101, Word to Mem = 110, Long to Mem = 111

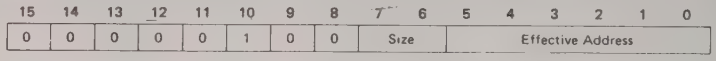
OR Immediate

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	Size		Effective Address					

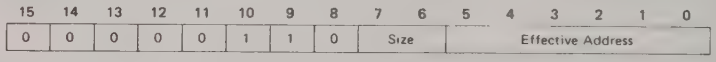
AND Immediate

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	1	0	Size		Effective Address					

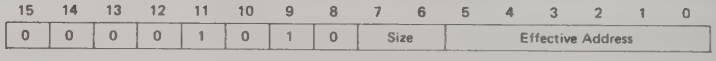
SUB Immediate



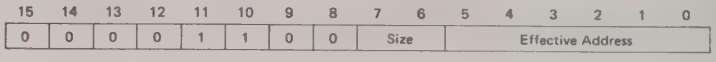
ADD Immediate



EOR Immediate

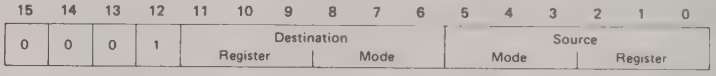


CMP Immediate



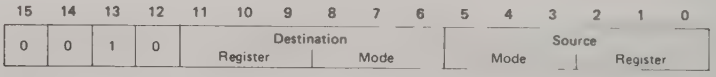
(2) MOVE BYTE INSTRUCTION

MOVE Byte



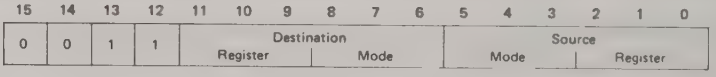
(3) MOVE LONG INSTRUCTION

MOVE Long



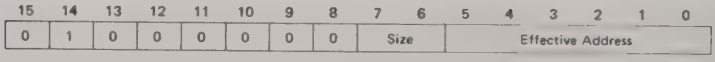
(4) MOVE WORD INSTRUCTION

MOVE Word

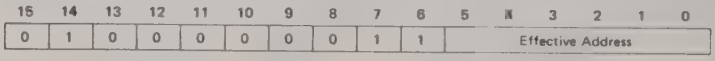


(5) MISCELLANEOUS INSTRUCTIONS

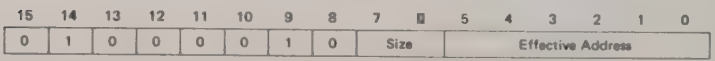
NEGX



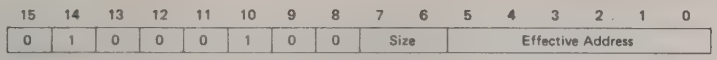
MOVE from SR



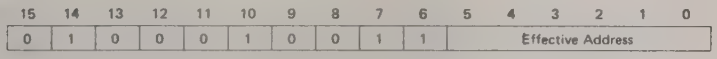
CLR



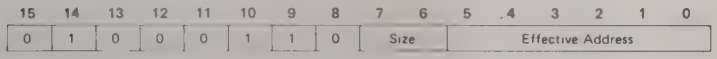
NEG



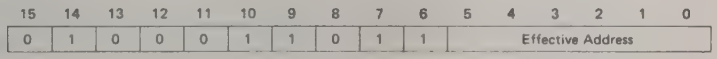
MOVE to CCR



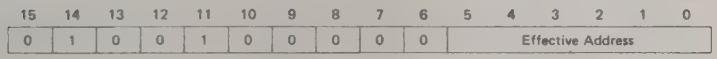
NOT



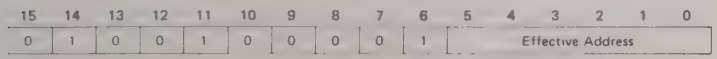
MOVE to SR



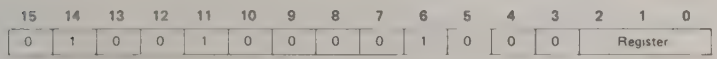
NBCD



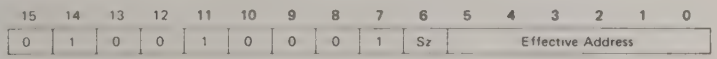
PEA



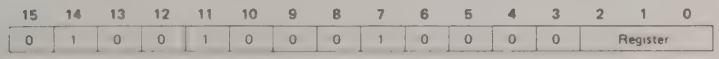
SWAP



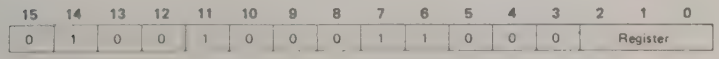
MOVEM Registers to EA



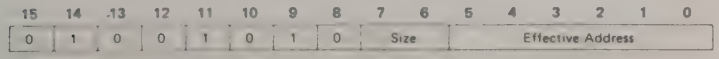
EXTW



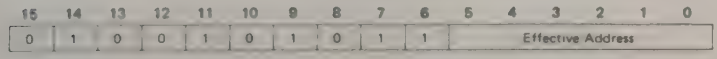
EXTL



TST



TAS



MOVEM EA to Registers

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	0	0	1	Sz	Effective Address					

TRAP

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	0	0	Vector			

LINK

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	0	1	0	Register		

UNLK

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	0	1	1	Register		

MOVE to USP

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	1	0	0	Register		

MOVE from USP

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	1	0	1	Register		

RESET

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	1	1	0	0	0	0

NOP

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	1	1	0	0	0	1

STOP

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	1	1	0	0	1	0

RTE

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	1	1	0	0	1	1

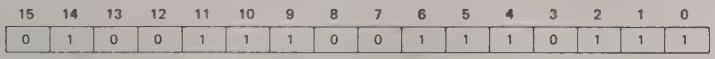
RTS

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	1	1	0	1	0	1

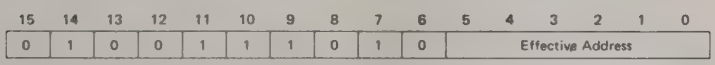
TRAPV

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	1	1	1	0	0	1	1	1	0	1	1	0

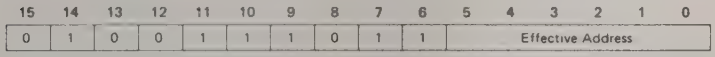
RTR



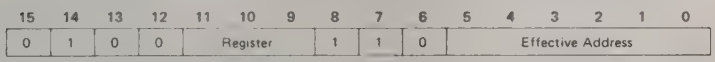
JSR



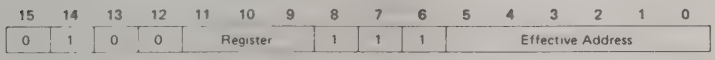
JMP



CHK

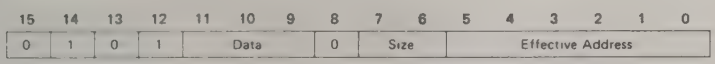


LEA



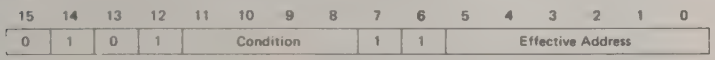
(6) ADD QUICK, SUBTRACT QUICK, SET CONDITIONALLY, DECREMENT INSTRUCTIONS

ADDQ

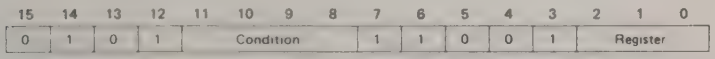


SUBQ

S_{cc}

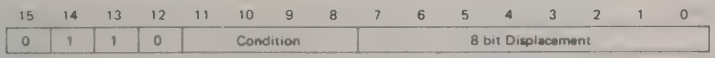


DB_{cc}

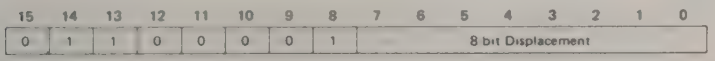


(7) BRANCH CONDITIONALLY, BRANCH TO SUBROUTINE INSTRUCTION

B_{cc}



BSR

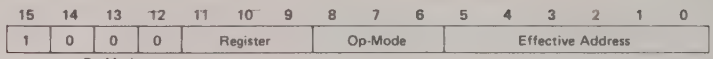


(8) MOVE QUICK INSTRUCTION

MOVEQ

(9) OR, DIVIDE, SUBTRACT DECIMAL INSTRUCTIONS

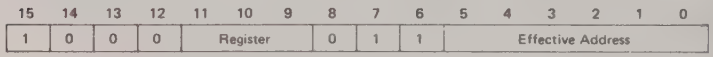
OR



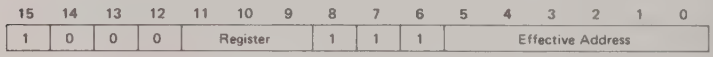
Op-Mode

B	W	L	
000	001	010	Dn ∨ EA → Dn
100	101	110	EA ∨ Dn → EA

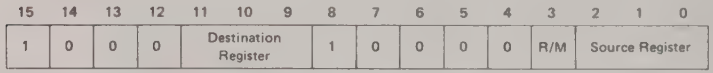
DIVU



DIVS



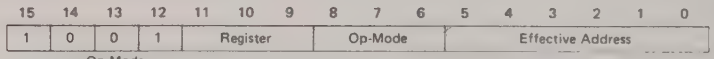
SBCD



R/M (register/memory): register – register = 0, memory – memory = 1

(10) SUBTRACT, SUBTRACT EXTENDED INSTRUCTIONS

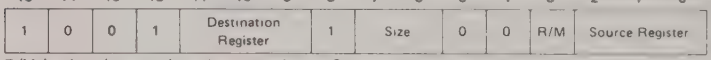
SUB



Op-Mode

B	W	L	
000	001	010	Dn–EA → Dn
100	101	110	EA–Dn → EA
–	011	111	An–EA → An

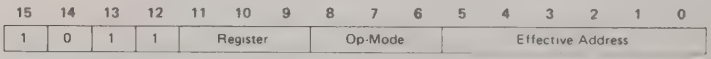
SUBX



R/M (register/memory): register – register = 0, memory – memory = 1

(11) COMPARE, EXCLUSIVE OR INSTRUCTIONS

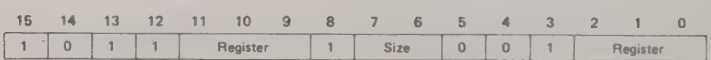
CMP



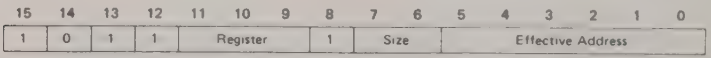
Op-Mode

B	W	L	
000	001	010	Dn–EA
–	011	111	An–EA

CMPM

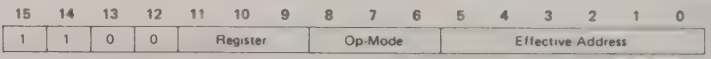


EOR



(12) AND, MULTIPLY, ADD DECIMAL, EXCHANGE INSTRUCTIONS

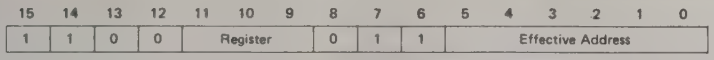
AND



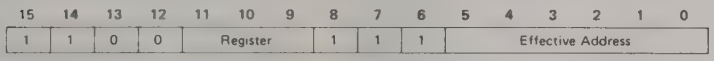
Op-Mode

B	W	L	
000	001	010	Dn ∧ EA → Dn
100	101	110	EA ∧ Dn → EA

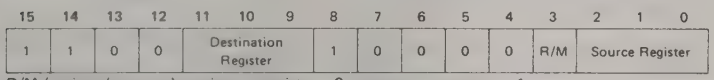
MULU



MULS

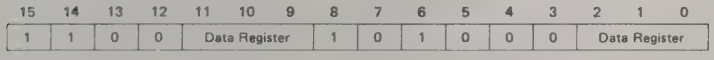


ABCD

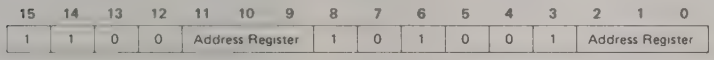


R/M (register/memory): register – register = 0, memory – memory = 1

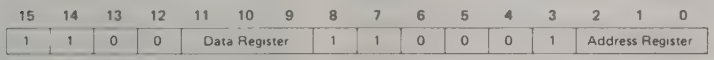
EXGD



EXGA

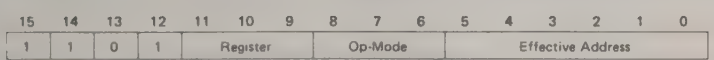


EXGM



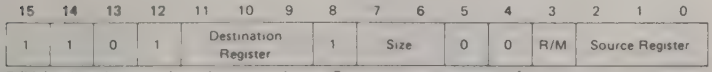
(13) ADD, ADD EXTENDED INSTRUCTIONS

ADD



Op-Mode			
B	W	L	
000	001	010	Dn + EA → Dn
100	101	110	EA + Dn → EA
–	011	111	An + EA → An

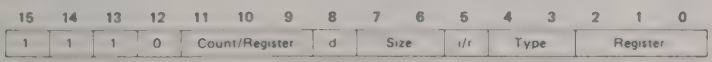
ADDX



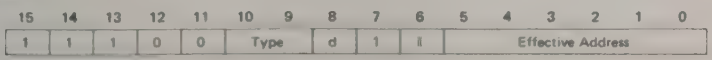
R/M (register/memory): register – register = 0, memory – memory = 1

(14) SHIFT/ROTATE INSTRUCTIONS

Data Register Shifts



Memory Shifts



Shift Type Codes: AS = 00, LS = 01, ROX = 10, RO = 11
d (direction): Right = 0, Left = 1
i/r (count source): Immediate Count = 0, Register Count = 1

■ INSTRUCTION EXECUTION TIMES

The following paragraphs contain listings of the instruction execution times in terms of external clock (CLK) periods. In this timing data, it is assumed that both memory read and write cycle times are four clock periods. Any wait states caused by a longer memory cycle must be added to the total instruction time. The number of bus read and write cycles for each instruction is also included with the timing data. This data is enclosed in parenthesis following the execution periods and is shown as: (r/w) where r is the number of read cycles and w is the number of write cycles.

(NOTE) The number of periods includes instruction fetch and all applicable operand fetches and stores.

• EFFECTIVE ADDRESS OPERAND CALCULATION TIMING

Table 27 lists the number of clock periods required to compute an instruction's effective address. It includes fetching of any extension words, the address computation, and fetching of the memory operand. The number of bus read and write cycles is shown in parenthesis as (r/w). Note there are no write cycles involved in processing the effective address.

• MOVE INSTRUCTION CLOCK PERIODS

Table 28 and 29 indicate the number of clock periods for the move instruction. This data includes instruction fetch, operand reads, and operand writes. The number of bus read and write cycles is shown in parenthesis as: (r/w).

• STANDARD INSTRUCTION CLOCK PERIODS

The number of clock periods shown in Table 30 indicates the time required to perform the operations, store the results, and read the next instruction. The number of bus read and write cycles is shown in parenthesis as: (r/w). The number of clock periods and the number of read and write cycles must be added respectively to those of the effective address calculation where indicated.

In Table 30 the headings have the following meanings: An = address register operand, Dn = data register operand, ea = an operand specified by an effective address, and M = memory effective address operand.

• IMMEDIATE INSTRUCTION CLOCK PERIODS

The number of clock periods shown in Table 31 includes the time to fetch immediate operands, perform the operations, store the results, and read the next operation. The number of bus read and write cycles is shown in parenthesis as: (r/w). The number of clock periods and the number of read and write cycles must be added respectively to those of the effective address calculation where indicated.

In Table 31, the headings have the following meanings: # = immediate operand, Dn = data register operand, An = address register operand, M = memory operand, CCR = condition code register, and SR = status register.

• SINGLE OPERAND INSTRUCTION CLOCK PERIODS

Table 32 indicates the number of clock periods for the single operand instructions. The number of bus read and write cycles is shown in parenthesis as: (r/w). The number of clock periods and the number of read and write cycles must be added respectively to those of the effective address calculation where indicated.

Table 27 Effective Address Calculation Timing

Addressing Mode		Byte, Word	Long
	Register		
Dn	Data Register Direct	0(0/0)	0(0/0)
An	Address Register Direct	0(0/0)	0(0/0)
	Memory		
An@	Address Register Indirect	4(1/0)	8(2/0)
An@ +	Address Register Indirect with Postincrement	4(1/0)	8(2/0)
An@ –	Address Register Indirect with Predecrement	6(1/0)	10(2/0)
An@(d)	Address Register Indirect with Displacement	8(2/0)	12(3/0)
An@(d, ix)*	Address Register Indirect with Index	10(2/0)	14(3/0)
xxx.W	Absolute Short	8(2/0)	12(3/0)
xxx.L	Absolute Long	12(3/0)	16(4/0)
PC@(d)	Program Counter with Displacement	8(2/0)	12(3/0)
PC@(d, ix)*	Program Counter with Index	10(2/0)	14(3/0)
#xxx	Immediate	4(1/0)	8(2/0)

* The size of the index register (ix) does not affect execution time.

Table 28 Move Byte and Word Instruction Clock Periods

Source	Destination								
	Dn	An	An@	An@ +	An@ –	An@(d)	An@(d, ix)*	xxx. W	xxx. L
Dn	4(1/0)	4(1/0)	8(1/1)	8(1/1)	8(1/1)	12(2/1)	14(2/1)	12(2/1)	16(3/1)
An	4(1/0)	4(1/0)	8(1/1)	8(1/1)	8(1/1)	12(2/1)	14(2/1)	12(2/1)	16(3/1)
An@	8(2/0)	8(2/0)	12(2/1)	12(2/1)	12(2/1)	16(3/1)	18(3/1)	16(3/1)	20(4/1)
An@+	8(2/0)	8(2/0)	12(2/1)	12(2/1)	12(2/1)	16(3/1)	18(3/1)	16(3/1)	20(4/1)
An@ -	10(2/0)	10(2/0)	14(2/1)	14(2/1)	14(2/1)	18(3/1)	20(3/1)	18(3/1)	22(4/1)
An@(d)	12(3/0)	12(3/0)	16(3/1)	16(3/1)	16(3/1)	20(4/1)	22(4/1)	20(4/1)	24(5/1)
An@(d, ix)*	14(3/0)	14(3/0)	18(3/1)	18(3/1)	18(3/1)	22(4/1)	24(4/1)	22(4/1)	26(5/1)
xxx. W	12(3/0)	12(3/0)	16(3/1)	16(3/1)	16(3/1)	20(4/1)	22(4/1)	20(4/1)	24(5/1)
xxx L	16(4/0)	16(4/0)	20(4/1)	20(4/1)	20(4/1)	24(5/1)	26(5/1)	24(5/1)	28(6/1)
PC@(d)	12(3/0)	12(3/0)	16(3/1)	16(3/1)	16(3/1)	20(4/1)	22(4/1)	20(4/1)	24(5/1)
PC@(d, ix)*	14(3/0)	14(3/0)	18(3/1)	18(3/1)	18(3/1)	22(4/1)	24(4/1)	22(4/1)	26(5/1)
#xxx	8(2/0)	8(2/0)	12(2/1)	12(2/1)	12(2/1)	16(3/1)	18(3/1)	16(3/1)	20(4/1)

* The size of the index register (ix) does not affect execution time.

Table 29 Move Long Instruction Clock Periods

Source	Destination								
	Dn	An	An@	An@ +	An@ –	An@(d)	An@(d, ix)*	xxx. W	xxx. L
Dn	4(1/0)	4(1/0)	12(1/2)	12(1/2)	12(1/2)	16(2/2)	18(2/2)	16(2/2)	20(3/2)
An	4(1/0)	4(1/0)	12(1/2)	12(1/2)	12(1/2)	16(2/2)	18(2/2)	16(2/2)	20(3/2)
An@	12(3/0)	12(3/0)	20(3/2)	20(3/2)	20(3/2)	24(4/2)	26(4/2)	24(4/2)	28(5/2)
An@+	12(3/0)	12(3/0)	20(3/2)	20(3/2)	20(3/2)	24(4/2)	26(4/2)	24(4/2)	28(5/2)
An@ -	14(3/0)	14(3/0)	22(3/2)	22(3/2)	22(3/2)	26(4/2)	28(4/2)	26(4/2)	30(5/2)
An@(d)	16(4/0)	16(4/0)	24(4/2)	24(4/2)	24(4/2)	28(5/2)	30(5/2)	28(5/2)	32(6/2)
An@(d, ix)*	18(4/0)	18(4/0)	26(4/2)	26(4/2)	26(4/2)	30(5/2)	32(5/2)	30(5/2)	34(6/2)
xxx. W	16(4/0)	16(4/0)	24(4/2)	24(4/2)	24(4/2)	28(5/2)	30(5/2)	28(5/2)	32(6/2)
xxx. L	20(5/0)	20(5/0)	28(5/2)	28(5/2)	28(5/2)	32(6/2)	34(6/2)	32(6/2)	36(7/2)
PC@(d)	16(4/0)	16(4/0)	24(4/2)	24(4/2)	24(4/2)	28(5/2)	30(5/2)	28(5/2)	32(6/2)
PC@(d, ix)*	18(4/0)	18(4/0)	26(4/2)	26(4/2)	26(4/2)	30(5/2)	32(5/2)	30(5/2)	34(6/2)
#xxx	12(3/0)	12(3/0)	20(3/2)	20(3/2)	20(3/2)	24(4/2)	26(4/2)	24(4/2)	28(5/2)

* The size of the index register (ix) does not affect execution time.

Table 30 Standard Instruction Clock Periods

Instruction	Size	op < ea >, An	op < ea >, Dn	op Dn, < M >
ADD	Byte, Word	8(1/0) +	4(1/0) +	8(1/1) +
	Long	6(1/0) + **	6(1/0) + **	12(1/2) +
AND	Byte, Word	–	4(1/0) +	8(1/1) +
	Long	–	6(1/0) + **	12(1/2) +
CMP	Byte, Word	6(1/0) +	4(1/0) +	–
	Long	6(1/0) +	6(1/0) +	–
DIVS	–	–	158(1/0) + *	–
DIVU	–	–	140(1/0) + *	–
EOR	Byte, Word	–	4(1/0) ***	8(1/1) +
	Long	–	8(1/0) ***	12(1/2) +
MULS	–	–	70(1/0) + *	–
MULU	–	–	70(1/0) + *	–
OR	Byte, Word	–	4(1/0) +	8(1/1) +
	Long	–	6(1/0) + **	12(1/2) +
SUB	Byte, Word	8(1/0) +	4(1/0) +	8(1/1) +
	Long	6(1/0) + **	6(1/0) + **	12(1/2) +

+ add effective address calculation time
* indicates maximum value
** total of 8 clock periods for instruction if the effective address is register direct
*** only available effective address mode is data register direct

- DIVS, DIVU – The divide algorithm used by the HD68000 provides less than 10% difference between the best and worst case timings
MULS, MULU – The multiply algorithm requires 38+2n clocks when n is defined as
MULU; n = the number of ones in the < ea >.
MULS; n = concatanate the < ea > with a zero as the LSB; n is the resultant number of 10 or 01 patterns in the 17-bit source; i.e. worst case happens when the source is $5555

Table 31 Immediation Instruction Clock Periods

Instruction	Size	op #, Dn	op #, An	op #, M	op #, CCR/SR
ADDI	Byte, Word	8(2/0)	–	12(2/1) +	–
	Long	16(3/0)	–	20(3/2) +	–
ADDQ	Byte, Word	4(1/0)	8(1/0)*	8(1/1) +	–
	Long	8(1/0)	8(1/0)	12(1/2) +	–
ANDI	Byte, Word	8(2/0)	–	12(2/1) +	20(3/0)
	Long	16(3/0)	–	20(3/1) +	–
CMPI	Byte, Word	8(2/0)	8(2/0)	8(2/0) +	–
	Long	14(3/0)	14(3/0)	12(3/0) +	–
EORI	Byte, Word	8(2/0)	–	12(2/1) +	20(3/0)
	Long	16(3/0)	–	20(3/2) +	–
MOVEQ	Long	4(1/0)	–	–	–
ORI	Byte, Word	8(2/0)	–	12(2/1) +	20(3/0)
	Long	16(3/0)	–	20(3/2) +	–
SUBI	Byte, Word	8(2/0)	–	12(2/1) +	–
	Long	16(3/0)	–	20(3/2) +	–
SUBQ	Byte, Word	4(1/0)	8(1/0)*	8(1/1) +	–
	Long	8(1/0)	8(1/0)	12(1/2) +	–

+ add effective address calculation time
* word only

Table 32 Single Operand Instruction Clock Periods

Instruction	Size	Register	Memory
CLR	Byte, Word	4(1/0)	8(1/1) +
	Long	6(1/0)	12(1/2) +
NBCD	Byte	6(1/0)	8(1/1) +
NEG	Byte, Word	4(1/0)	8(1/1) +
	Long	6(1/0)	12(1/2) +
NEGX	Byte, Word	4(1/0)	8(1/1) +
	Long	6(1/0)	12(1/2) +
NOT	Byte, Word	4(1/0)	8(1/1) +
	Long	6(1/0)	12(1/2) +
S_{cc}	Byte, False	4(1/0)	8(1/1) +
	Byte, True	6(1/0)	8(1/1) +
TAS	Byte	4(1/0)	10(1/1) +
TST	Byte, Word	4(1/0)	4(1/0) +
	Long	4(1/0)	4(1/0) +

+ add effective address calculation time

• SHIFT/ROTATE INSTRUCTION CLOCK PERIODS

Table 33 indicates the number of clock periods for the shift and rotate instructions. The number of bus read and write cycles is shown in parenthesis as: (r/w). The number of clock periods and the number of read and write cycles must be added respectively to those of the effective address calculation where indicated.

• BIT MANIPULATION INSTRUCTION CLOCK PERIODS

Table 34 indicates the number of clock periods required for the bit manipulation instructions. The number of bus read and write cycles is shown in parenthesis as: (r/w). The number of clock periods and the number of read and write cycles must be added respectively to those of the effective address calculation where indicated.

- **CONDITIONAL INSTRUCTION CLOCK PERIODS**

Table 35 indicates the number of clock periods required for the conditional instructions. The number of bus read and write cycles is indicated in parenthesis as: (r/w). The number of clock periods and the number of read and write cycles must be added respectively to those of the effective address calculation where indicated.

- **JMP, JSR, LEA, PEA, MOVEM INSTRUCTION CLOCK PERIODS**

Table 36 indicates the number of clock periods required for the jump, jump to subroutine, load effective address, push effective address, and move multiple registers instructions. The number of bus read and write cycles is shown in parenthesis as: (r/w).

Table 33 Shift/Rotate Instruction Clock Periods

Instruction	Size	Register	Memory
ASR, ASL	Byte, Word	6 + 2n(1/0)	8(1/1) +
	Long	8 + 2n(1/0)	–
LSR, LSL	Byte, Word	6 + 2n(1/0)	8(1/1) +
	Long	8 + 2n(1/0)	–
ROR, ROL	Byte, Word	6 + 2n(1/0)	8(1/1) +
	Long	8 + 2n(1/0)	–
ROXR, ROXL	Byte, Word	6 + 2n(1/0)	8(1/1) +
	Long	8 + 2n(1/0)	–

Table 34 Bit Manipulation Instruction Clock Periods

Instruction	Size	Dynamic		Static	
		Register	Memory	Register	Memory
BCHG	Byte	–	8(1/1) +	–	12(2/1) +
	Long	8(1/0)*	–	12(2/0)*	–
BCLR	Byte	–	8(1/1) +	–	12(2/1) +
	Long	10(1/0)*	–	14(2/0)*	–
BSET	Byte	–	8(1/1) +	–	12(2/1) +
	Long	8(1/0)*	–	12(2/0)*	–
BTST	Byte	–	4(1/0) +	–	8(2/0) +
	Long	6(1/0)	–	10(2/0)	–

+ add effective address calculation time
* indicates maximum value

Table 35 Conditional Instruction Clock Periods

Instruction	Displacement	Trap or Branch Taken	Trap of Branch Not Taken
B_{cc}	Byte	10(2/0)	8(1/0)
	Word	10(2/0)	12(2/0)
BRA	Byte	10(2/0)	–
	Word	10(2/0)	–
BSR	Byte	18(2/2)	–
	Word	18(2/2)	–
DB_{cc}	CC true	–	12(2/0)
	CC false	10(2/0)	14(3/0)
CHK	–	40(5/3) + *	10(1/0) +
TRAP	–	34(4/3)	–
TRAPV	–	34(5/3)	4(1/0)

+ add effective address calculation time
* indicates maximum value

Table 36 JMP, JSR, LEA, PEA, MOMEM Instruction Clock Periods

Instr	Size	An@	An@ +	An@ -	An@(d)	An@(d, ix)*	xxx.W	xxx.L	PC@(d)	PC@(d, ix)*
JMP	–	8(2/0)	–	–	10(2/0)	14(3/0)	10(2/0)	12(3/0)	10(2/0)	14(3/0)
JSR	–	16(2/2)	–	–	18(2/2)	22(2/2)	18(2/2)	20(3/2)	18(2/2)	22(2/2)
LEA	–	4(1/0)	–	–	8(2/0)	12(2/0)	8(2/0)	12(3/0)	8(2/0)	12(2/0)
PEA	–	12(1/2)	–	–	16(2/2)	20(2/2)	16(2/2)	20(3/2)	16(2/2)	20(2/2)
MOVEM	Word	12+4n (3+n/0)	12+4n (3+n/0)	– –	16+4n (4+n/0)	18+4n (4+n/0)	16+4n (4+n/0)	20+4n (5+n/0)	16+4n (4+n/0)	18+4n (4+n/0)
M → R	Long	12+8n (3+2n/0)	12+8n (3+2n/0)	– –	16+8n (4+2n/0)	18+8n (4+2n/0)	16+8n (4+2n/0)	20+8n (5+2n/0)	16+8n (4+2n/0)	18+8n (4+2n/0)
MOVEM	Word	8+4n (2/n)	– –	8+4n (2/n)	12+4n (3/n)	14+4n (3/n)	12+4n (3/n)	16+4n (4/n)	– –	– –
R → M	Long	8+8n (2/2n)	– –	8+8n (2/2n)	12+8n (3/2n)	14+8n (3/2n)	12+8n (3/2n)	16+8n (4/2n)	– –	– –

n is the number of registers to move

* is the size of the index register (ix) does not affect the instruction's execution time

• MULTI-PRECISION INSTRUCTION CLOCK PERIODS

Table 37 indicates the number of clock periods for the multi-precision instructions. The number of clock periods includes the time to fetch both operands, perform the operations, store the results, and read the next instructions. The number of read and write cycles is shown in parenthesis as: (r/w).

In Table 37, the headings have the following meanings: Dn = data register operand and M = memory operand.

Table 37 Multi-Precision Instruction Clock Periods

Instruction	Size	op Dn, Dn	op M, M
ADDX	Byte, Word	4(1/0)	18(3/1)
	Long	8(1/0)	30(5/2)
CMPM	Byte, Word	–	12(3/0)
	Long	–	20(5/0)
SUBX	Byte, Word	4(1/0)	18(3/1)
	Long	8(1/0)	30(5/2)
ABCD	Byte	6(1/0)	18(3/1)
SBCD	Byte	6(1/0)	18(3/1)

• MISCELLANEOUS INSTRUCTION CLOCK PERIODS

Table 38 indicates the number of clock periods for the following miscellaneous instructions. The number of bus read and write cycles is shown in parenthesis as: (r/w). The number of clock periods plus the number of read and write cycles must be added to those of the effective address calculation where indicated.

• EXCEPTION PROCESSING CLOCK PERIODS

Table 39 indicates the number of clock periods for exception processing. The number of clock periods includes the time for all stacking, the vector fetch, and the fetch of the first instruction of the handler routine. The number of bus read and write cycles is shown in parenthesis as: (r/w).

Table 38 Miscellaneous Instruction Clock Periods

Instruction	Size	Register	Memory	Register → Memory	Memory → Register
MOVE from SR	–	6(1/0)	8(1/1) +	–	–
MOVE to CCR	–	12(2/0)	12(2/0) +	–	–
MOVE to SR	–	12(2/0)	12(2/0) +	–	–
MOVEP	Word	–	–	16(2/2)	16(4/0)
	Long	–	–	24(2/4)	24(6/0)
EXG	–	6(1/0)	–	–	–
EXT	Word	4(1/0)	–	–	–
	Long	4(1/0)	–	–	–
LINK	–	16(2/2)	–	–	–
MOVE from USP	–	4(1/0)	–	–	–
MOVE to USP	–	4(1/0)	–	–	–
NOP	–	4(1/0)	–	–	–
RESET	–	132(1/0)	–	–	–
RTE	–	20(5/0)	–	–	–
RTR	–	20(5/0	–	–	–
RTS	–	16(4/0)	–	–	–
STOP	–	4(0/0)	–	–	–
SWAP	–	4(1/0)	–	–	–
UNLK	–	12(3/0)	–	–	–

+ add effective address calculation time

Table 39 Exception Processing Clock Periods

Exception	Periods
Reset**	38.5 (6/10)
Address Error	50(4/7)
Bus Error	50(4/7)
Interrupt	44(5/3)*
Illegal Instruction	34(4/3)
Privileged Violation	34(4/3)
Trace	34(4/3)

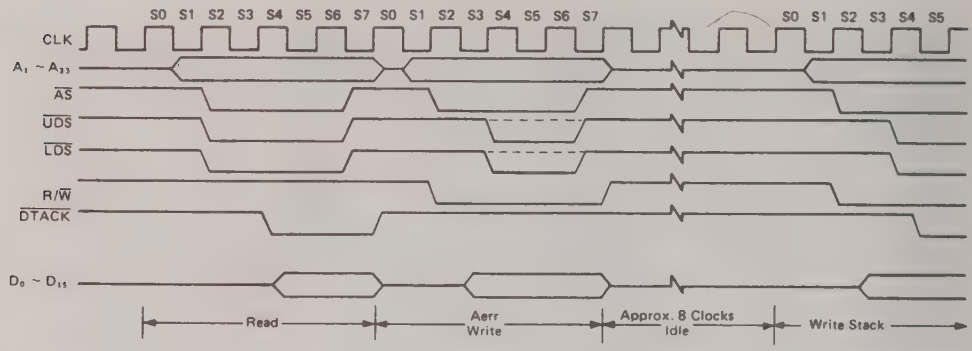
* The interrupt acknowledge bus cycle is assumed to take four external clock periods.

** Indicates the time from when $\overline{RES}$ and $\overline{HALT}$ are first sampled as negated to when instruction execution starts.

■ MASK VERSION

Type No.	Mask version
HD68000-6 HD68000-8 HD68000-10 HD68000-12	68000S1
HD68000Y6 HD68000Y8 HD68000Y10 HD68000Y12	
HD68000P6 HD68000P8 HD68000PS6 HD68000PS8	68000U

The difference of function between mask version 68000S1 and 68000U is only as following (Figure 57).



Broken line: mask version 68000U

Figure 57 Address Error Timing

HD68450, HD68450Y
DMAC (Direct Memory Access Controller)

Microprocessor implemented systems are becoming increasingly complex, particularly with the advent of high-performance 16-bit MPU devices with large memory addressing capability. In order to maintain high throughput, large blocks of data must be moved within these systems in a quick, efficient manner with minimum intervention by the MPU itself.

The HD68450 Direct Memory Access Controller (DMAC) is designed specifically to complement the performance and architectural capabilities of the HD68000 MPU by providing the following features:

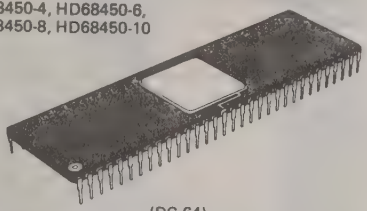
- HMCS68000 Bus Compatible
- 4 independent DMA Channels
- Memory-to-Memory, Memory-to-Device, Device-to-Memory Transfers
- MMU Compatible
- Array-Chained and Linked-Array-Chained Operations
- On-Chip Registers that allow Complete Software Control by the System MPU
- Interface Lines for Requesting, Acknowledging, and Incidental Control of the Peripheral Devices
- Variable System Bus Bandwidth Utilization
- Programmable Channel Prioritization
- 2 Vectored interrupts for each Channel
- Auto-Request and External-Request Transfer Modes
- +5 Volt Operation

The DMAC functions by transferring a series of operands (data) between memory and peripheral device; operand sizes can be byte, word, or long word. A block is a sequence of operations; the number of operands in a block is determined by a transfer count. A single-channel operation may involve the transfer of several blocks of data between memory and device.

■ TYPE OF PRODUCTS

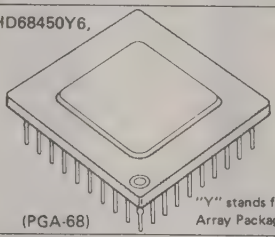
Type No.	Bus Timing	Packaging
HD68450-4	4MHz	DC-64
HD68450-6	6MHz	
HD68450-8	8MHz	
HD68450-10	10MHz	
HD68450Y4	4MHz	PGA-68
HD68450Y6	6MHz	
HD68450Y8	8MHz	
HD68450Y10	10MHz	

HD68450-4, HD68450-6,
HD68450-8, HD68450-10



(DC-64)

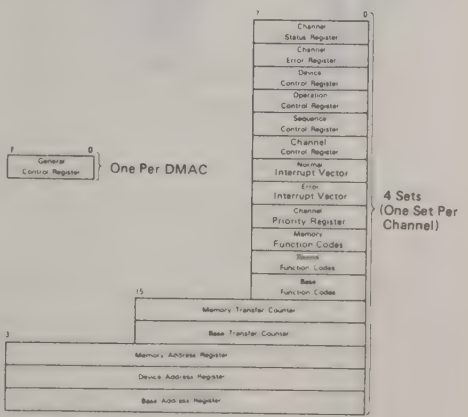
HD68450Y4, HD68450Y6,
HD68450Y8,
HD68450Y10



(PGA-68)

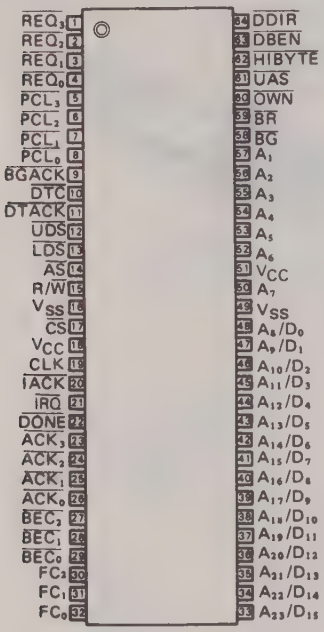
"Y" stands for Pin Grid Array Package.

■ PROGRAMMING MODEL



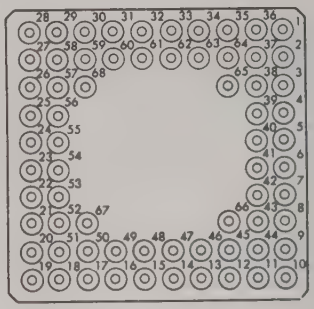
■ PIN ARRANGEMENT

● HD68450



(Top View)

● HD68450Y



(Bottom View)

Pin No.	Function	Pin No.	Function	Pin No.	Function	Pin No.	Function
1	N/C	18	$\overline{PCL_1}$	35	A_{19}/D_{11}	52	$\overline{BGACK}$
2	A_{13}/D_5	19	$\overline{DTACK}$	36	A_{17}/D_9	53	$\overline{LDS}$
3	A_{11}/D_3	20	$\overline{UDS}$	37	A_{15}/D_7	54	V_{SS}
4	A_{10}/D_2	21	$\overline{AS}$	38	A_{12}/D_4	55	V_{CC}
5	A_8/D_0	22	R/$\overline{W}$	39	A_9/D_1	56	$\overline{DONE}$
6	A_7	23	N/C	40	V_{SS}	57	$\overline{IRQ}$
7	A_6	24	$\overline{CS}$	41	V_{CC}	58	$\overline{ACK_2}$
8	A_5	25	CLK	42	A_4	59	$\overline{BEC_2}$
9	A_3	26	$\overline{IACK}$	43	A_2	60	$\overline{BEC_0}$
10	N/C	27	$\overline{ACK_3}$	44	$\overline{BG}$	61	FC_0
11	$\overline{BR}$	28	$\overline{ACK_0}$	45	$\overline{OWN}$	62	A_{21}/D_{13}
12	$\overline{UAS}$	29	$\overline{BEC_1}$	46	HIBYTE	63	A_{18}/D_{10}
13	$\overline{DBEN}$	30	FC_2	47	$\overline{DDIR}$	64	A_{16}/D_8
14	$\overline{REQ_3}$	31	FC_1	48	$\overline{REQ_1}$	65	A_{14}/D_6
15	$\overline{REQ_2}$	32	A_{23}/D_{15}	49	$\overline{PCL_2}$	66	A_1
16	$\overline{REQ_0}$	33	A_{22}/D_{14}	50	$\overline{PCL_0}$	67	$\overline{DTC}$
17	$\overline{PCL_3}$	34	A_{20}/D_{12}	51	N/C	68	$\overline{ACK_1}$

■ ABSOLUTE MAXIMUM RATINGS

Item	Symbol	Value	Unit
Supply Voltage	V_{CC}*	−0.3 ~ +7.0	V
Input Voltage	V_{in}*	−0.3 ~ +7.0	V
Operating Temperature Range	T_{opr}	0 ~ +70	°C
Storage Temperature	T_{stg}	−55 ~ +150	°C

* With respect to V_{SS} (SYSTEM GND)

(NOTE) Permanent LSI damage may occur if maximum ratings are exceeded. Normal operation should be under recommended operating conditions. If these conditions are exceeded, it could affect reliability of LSI.

■ RECOMMENDED OPERATING CONDITIONS

Item	Symbol	min	typ	max	Unit
Supply Voltage	V_{CC}*	4.75	5.0	5.25	V
Input Voltage	V_{IH}*	2.0	–	V_{CC}	V
	V_{IL}*	−0.3	–	0.8	V
Operating Temperature	T_{opr}	0	25	70	°C

* With respect to V_{SS} (SYSTEM GND)

■ ELECTRICAL CHARACTERISTICS

● DC CHARACTERISTICS (V_{CC} = 5V ±5%, V_{SS} = 0V, Ta = 0 ~ +70°C, unless otherwise noted.)

Item		Symbol	Test Condition	min	typ	max	Unit
Input "High" Voltage		V_{IH}		2.0	–	V_{CC}	V
Input "Low" Voltage		V_{IL}		V_{SS} −0.3	–	0.8	V
Input Leakage Current	$\overline{CS}$, $\overline{IACK}$, $\overline{BG}$, CLK, $\overline{BEC_0} \sim \overline{BEC_2}$, $\overline{REQ_0} \sim \overline{REQ_3}$	I_{in}		–	–	10	μA
Three-State (Off State) Input Current	$\overline{A_1} \sim \overline{A_7}$, $D_0 \sim D_{15}/A_8 \sim A_{23}$, $\overline{AS}$, $\overline{UDS}$, $\overline{LDS}$, R/$\overline{W}$, $\overline{UAS}$, $\overline{DTACK}$, $\overline{BGACK}$, $\overline{OWN}$, $\overline{DTC}$, $\overline{HIBYTE}$, $\overline{DDIR}$, $\overline{DBEN}$, $FC_0 \sim FC_2$	I_{TSI}		–	–	10	μA
Open Drain (Off State) Input Current	$\overline{IRQ}$, $\overline{DONE}$	I_{ODI}		–	–	20	μA
Output "High" Voltage	$\overline{A_1} \sim \overline{A_7}$, $D_0 \sim D_{15}/A_8 \sim A_{23}$, $\overline{AS}$, $\overline{UDS}$, $\overline{LDS}$, R/$\overline{W}$, $\overline{UAS}$, $\overline{DTACK}$, $\overline{BGACK}$, $\overline{BR}$, $\overline{OWN}$, $\overline{DTC}$, $\overline{HIBYTE}$, $\overline{DDIR}$, $\overline{DBEN}$, $\overline{ACK_0} \sim \overline{ACK_3}$, $\overline{PCL_0} \sim \overline{PCL_3}$, $FC_0 \sim FC_2$	V_{OH}	I_{OH} = −400 μA	2.4	–	–	V
Output "Low" Voltage	$\overline{A_1} \sim \overline{A_7}$, $FC_0 \sim FC_2$	V_{OL}	I_{OL} = 3.2 mA	–	–	0.5	V
	$D_0 \sim D_{15}/A_8 \sim A_{23}$, $\overline{AS}$, $\overline{UDS}$, $\overline{LDS}$, R/$\overline{W}$, $\overline{DTACK}$, $\overline{BR}$, $\overline{OWN}$, $\overline{DTC}$, $\overline{HIBYTE}$, $\overline{DDIR}$, $\overline{DBEN}$, $\overline{ACK_0} \sim \overline{ACK_3}$, $\overline{UAS}$, $\overline{PCL_0} \sim \overline{PCL_3}$, $\overline{BGACK}$	V_{OL}	I_{OL} = 5.3 mA	–	–	0.5	
	$\overline{IRQ}$, $\overline{DONE}$	V_{OL}	I_{OL} = 8.9 mA	–	–	0.5	
Power Dissipation		P_D	f = 8 MHz, V_{CC} = 5.0 V, Ta = 25°C	–	1.4	2.0	W
Capacitance		C_{in}	V_{in} = 0V, Ta = 25°C, f = 1 MHz	–	–	15	pF

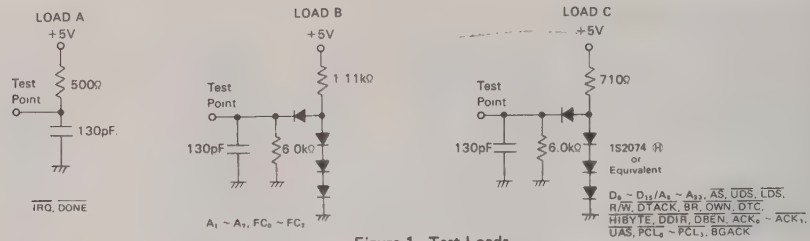


Figure 1 Test Loads

• AC ELECTRICAL SPECIFICATIONS (V_{CC} = 5V ±5%, V_{SS} = 0V, Ta = 0~+70°C)

No	Item	Symbol	Test Condition	4MHz HD68450-4 HD68450Y4 min	4MHz HD68450-4 HD68450Y4 max	6MHz HD68450-6 HD68450Y6 min	6MHz HD68450-6 HD68450Y6 max	8MHz HD68450-8 HD68450Y8 min	8MHz HD68450-8 HD68450Y8 max	10MHz HD68450-10 HD68450Y10 min	10MHz HD68450-10 HD68450Y10 max	Unit
	Frequency of Operation	f		2	4	2	6	2	8	2	10	MHz
1	Clock Period	t_{cyc}		250	500	167	500	126	500	100	500	ns
2	Clock Width Low	t_{CL}		115	250	75	250	55	250	45	250	ns
3	Clock Width High	t_{CH}		115	250	75	250	55	250	45	250	ns
4	Clock Fall Time	t_{Cf}		–	10	–	10	–	10	–	10	ns
5	Clock Rise Time	t_{Cr}		–	10	–	10	–	10	–	10	ns
6	Asynchronous Input Setup Time	t_{ASI}		30	–	25	–	20	–	15		ns
7	Data in to DBEN Low	t_{DIDBL}		0	–	0	–	0	–	0	–	ns
8	DTACK Low to Data Invalid	t_{DTLDI}		0	–	0	-	0	–	0		ns
9	Address in to AS in Low	t_{AIASL}		0	–	0	–	0	–	0	-	ns
10	AS, DS in High to Address in Invalid	t_{SIHAIV}		0	–	0	–	0	–	0	–	ns
10A	DS in High to CS High	t_{DSHCSH}		–	1 0	–	1 0	–	1.0	-	1 0	clk per
11	Clock High to DDIR Low	t_{CHDRL}		–	90		80	–	70		60	ns
12	Clock High to DDIR High	t_{CHDRH}		–	90	–	80	–	70	–	60	ns
13	DS in High to DDIR High Impedance	t_{DSHDRZ}			160	–	140	–	120	–	110	ns
14	Clock Low to DBEN Low	t_{CLDBL}		-	90	–	80	-	70		60	ns
15	Clock Low to DBEN High	t_{CLDBH}		–	90	–	80	–	70	–	60	ns
16	DS in High to DBEN High Impedance	t_{DSHDBZ}		–	160		140	-	120	–	110	ns
17	Clock High to Data Out Valid (MPU read)	t_{CHDVM}		–	290	–	230	–	180	–	160	ns
18	DS in High to Data Out Invalid	t_{DSHDZn}		0	–	0	–	0	–	0		ns
19	DS in High to Data High Impedance	t_{DSHDZ}		–	160	–	140	–	120	–	110	ns
20	Clock Low to DTACK Low	t_{CLDTL}		–	90	–	80	–	70	–	60	ns
21	DS in High to DTACK High	t_{DSHDTH}		-	160	–	130		110		110	ns
22	DTACK Width High	t_{DTH}		10	–	10		10		10	-	ns
23	DS in High to DTACK High Impedance	t_{DSHDTZ}			220	–	200	-	180		160	ns
24	DTACK Low to DS in High	t_{DTLDSH}		0	–	0	–	0	–	0	-	ns
25	REQ Width Low	t_{REQL}		2.0	–	2.0	–	2.0	–	2.0	–	clk per
26	REQ Low to BR Low	t_{RELBRL}		500	–	334	–	250	–	200	–	ns
27	Clock High to BR Low	t_{CHBRL}	Fig 1 ~ Fig. 8		90	–	80		70		60	ns
28	Clock High to BR High	t_{CHBRH}		–	90	–	80	–	70	–	60	ns
29	BG Low to BGACK Low	t_{BGLBL}		4 5		4 5	–	4 5	-	4 5	–	clk per
30	BR Low to MPU Cycle End (AS in High)	t_{BRLASH}		0	–	0	–	0	–	0	–	ns
31	MPU Cycle End (AS in High) to BGACK Low	t_{ASHBL}		4 5	5 5	4 5	5 5	4 5	5 5	4 5	5 5	clk per
32	REQ Low to BGACK Low	t_{REQLBL}		1 2 0	-	1 2 0	–	1 2 0		1 2 0	-	clk per
33	Clock High to BGACK High	t_{CHBL}		–	90	–	80	–	70	–	60	ns
34	Clock High to BGACK High	t_{CHBH}		–	90	–	80	–	70	–	60	ns
35	Clock Low to BGACK High Impedance	t_{CLBZ}		–	120	–	100	–	80	–	70	ns
36	Clock High to FC Valid	t_{CHFCV}		–	140	–	120	–	100	–	90	ns
37*	Clock High to Address Valid	t_{CHAV}		–	160	–	140	–	120	–	110	ns
38	Clock High to Address/FC/Data High Impedance	t_{CHAZx}		–	140	–	120		100		100	ns
39	Clock High to Address/FC/Data Invalid	t_{CHAZn}		0	–	0	–	0	-	0		ns
40	Clock Low to Address High Impedance	t_{CLAZ}		–	140	–	120		100		90	ns
41	Clock High to UAS Low	t_{CHUL}		–	90	–	80	–	70	–	60	ns
42	Clock High to UAS High	t_{CHUH}		–	90	-	80		70		60	ns
43	Clock Low to UAS High Impedance	t_{CLUZ}		–	120	–	100	–	80	–	70	ns
44	UAS High to Address Invalid	t_{UHAI}		50	–	40	–	30	–	20	–	ns
45	Clock High to AS, DS Low	t_{CHSL}		–	80	–	70	–	60	–	55	ns
46	Clock Low to DS Low (write)	t_{CLDSL}		-	80	–	70	-	60		55	ns
47	Clock Low to AS, DS High	t_{CLSH}		–	90	–	80	–	70	–	60	ns
48	Clock Low to AS, DS High Impedance	t_{CLSZ}		–	120	-	100	-	80		70	ns
49	AS Width Low	t_{ASL}		545	–	350	–	255	–	195	–	ns
50	DS Width Low	t_{DSL}		420	–	265		190	-	145	-	ns
51	AS, DS Width High	t_{SH}		285		180		150		105	-	ns
52	Address/FC Valid to AS, DS Low	t_{AVSL}		50	–	40	–	30	–	20		ns
53	AS, DS High to Address/FC/Data Invalid	t_{SHAZ}		50	–	40	–	30	-	20		ns
54	Clock High to R/W Low	t_{CHRL}		–	90	-	80		70		60	ns
55	Clock High to R/W High	t_{CHRH}		–	90	–	80	-	70		60	ns

(to be continued)

No	Item	Symbol	Test Condition	4MHz HD68450-4 HD68450Y4		6MHz HD68450-6 HD68450Y6		8MHz HD68450-8 HD68450Y8		10MHz HD68450-10 HD68450Y10		Unit
				min	max	min	max	min	max	min	max	
56	Clock Low to R/W High Impedance	t_{CLRZ}	Fig. 1 ~ Fig. 8	–	120	–	100	–	80	–	70	ns
57	Address/FC Valid to R/W Low	t_{AVRL}		100	–	40	–	20	–	10	–	ns
58	R/W Low to DS Low (write)	t_{RLSL}		285	–	170	–	120	–	90	–	ns
59	DS High to R/W High	t_{SHRH}		60	–	50	–	40	–	20	-	ns
60	Clock Low to OWN Low	t_{CLOL}		–	90		80		70		60	ns
61	Clock Low to OWN High	t_{CLOH}		–	90	–	80	–	70	–	60	ns
62	Clock High to OWN High Impedance	t_{CHOZ}		–	120	-	100	–	80	–	70	ns
63	OWN Low to BGACK Low	t_{OLBL}		50	–	40		30	–	20		ns
64	BGACK High to OWN High	t_{BHOH}		50	–	40	–	30	–	20		ns
65	OWN Low to UAS Low	t_{OLUL}		50	–	40	–	30	–	20	–	ns
66	Clock High to ACK Low	t_{CHACL}		–	90	–	80	–	70	–	60	ns
67	Clock Low to ACK Low	t_{CLACL}		-	90		80	–	70		60	ns
68	Clock High to ACK High	t_{CHACH}			90		80	-	70		60	ns
69	ACK Low to DS Low	t_{ACLDSL}		230	–	140	–	100	–	80		ns
70	DS High to ACK High	t_{DSHACH}		50	-	40	–	30	–	20		ns
71	Clock High to HIBYTE Low	t_{CHHIL}		–	90	–	80	–	70	–	60	ns
72	Clock Low to HIBYTE Low	t_{CLHIL}		–	90	–	80	–	70	–	60	ns
73	Clock High to HIBYTE High	t_{CHHIH}			90		80	–	70		60	ns
74	Clock Low to HIBYTE High Impedance	t_{CLHIZ}		–	120	–	100	–	80	–	70	ns
75	Clock High to DTC Low	t_{CHDTL}		–	90	–	80	–	70	–	60	ns
76	Clock High to DTC High	t_{CHDTH}		–	90	–	80	–	70	–	60	ns
77	Clock Low to DTC High Impedance	t_{CLDTZ}		–	120	-	100	–	80	-	70	ns
78	DTC Width Low	t_{DTCL}		230	–	147	–	105	–	80	–	ns
79	DTC Low to DS High	t_{DTLDH}		95	–	50	–	30	–	20	–	ns
80	Clock High to DONE Low	t_{CHDOL}		–	90	–	80	–	70	–	60	ns
81	Clock Low to DONE Low	t_{CLDOL}		–	90	–	80	–	70	–	60	ns
82	Clock High to DONE High	t_{CHDOH}		-	150		140		130	–	120	ns
83	Clock Low to DDIR High Impedance	t_{CLDRZ}		–	120	–	100	–	80	–	70	ns
84	Clock Low to DBEN High Impedance	t_{CLDBZ}		-	120	-	100	-	80	-	70	ns
85	DDIR Low to DBEN Low	t_{DRLDBL}		50		40		30		20	-	ns
86	DBEN High to DDIR High	t_{DBHDRH}		50	-	40	-	30		20		ns
87	DBEN Low to Address/Data High Impedance	t_{DBLAZ}			17		17		17	–	17	ns
88	Clock Low to PCL Low (1/8 clock)	t_{CLPL}		–	90	–	80	–	70	–	60	ns
89	Clock Low to PCL High (1/8 clock)	t_{CLPH}		–	90		80	–	70		60	ns
90	PCL Width Low (1/8 clock)	t_{PCLL}		4 0	-	4 0	–	4 0	-	4 0	–	clk per
91	DTACK Low to Data In (setup time)	t_{DALDI}			320	-	200		150	-	115	ns
92	DS High to Data Invalid (hold time)	t_{SHDI}		0		0	–	0		0		ns
93	DS High to DTACK High	t_{SHDAH}		0	240	0	160	0	120	0	90	ns
94	Data Out Valid to DS Low	t_{DOSL}		0	-	0	–	0		0	–	ns
95	Data In to Clock Low (setup time)	t_{DICL}		30	–	25	–	15	-	15	–	ns
96	BEC Low to DTACK Low	t_{BECDAL}		50	–	50	–	50	-	50	–	ns
97	BEC Width Low	t_{BECL}		2 0	-	2 0	-	2 0	–	2 0	-	clk per
98	Clock High to IRQ Low	t_{CHIRL}			90		80		70	-	60	ns
99	Clock High to IRQ High	t_{CHIRH}		–	150	–	140	–	130	–	120	ns
100	READY In to DTC Low (Read)	t_{RALDTL}		270	–	180	–	145	–	120	–	ns
101	READY In to DS Low (Write)	t_{RALDSL}		395	–	240	–	205	–	170	–	ns
102	DS High to READY High	t_{DSHRAH}		0	240	0	160	0	120	0	90	ns
103	DONE In Low to DTACK Low	t_{DOLDAL}		50		50		50		50	–	ns
104	DS High to DONE In High	t_{DSHDOH}		0	240	0	160	0	120	0	90	ns
105	Asynchronous Input Hold Time	t_{ASIH}		15		15	-	15		15	-	ns

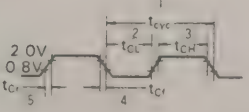


Figure 2 Input Clock Waveform

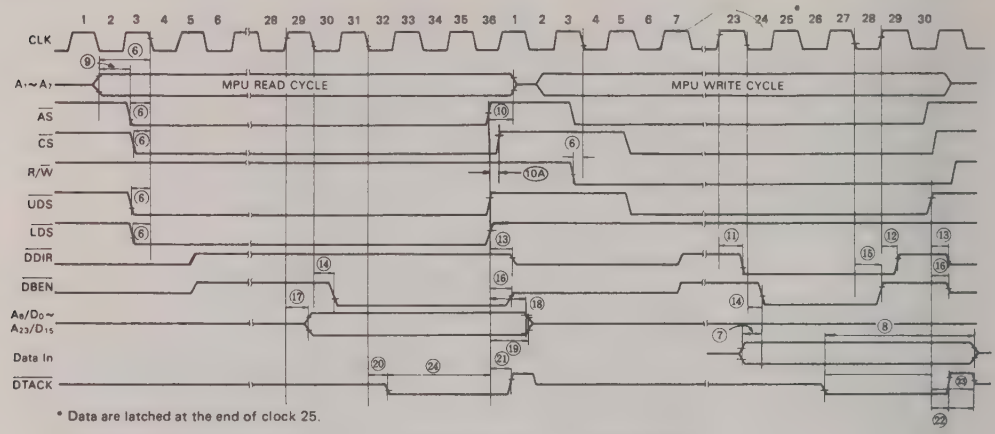


Figure 3 AC Electrical Waveforms — MPU Read/Write

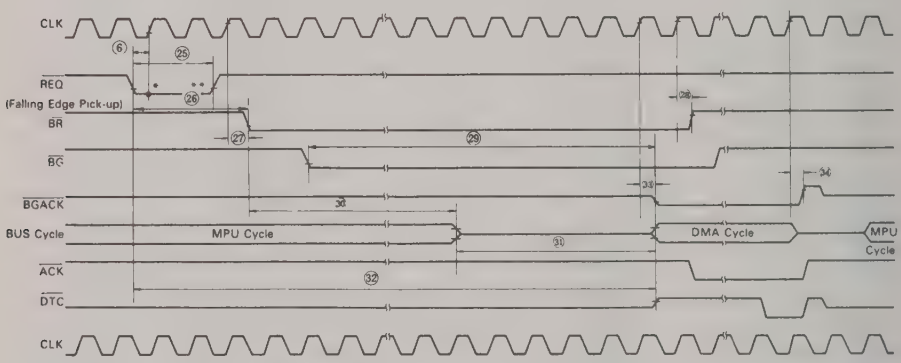
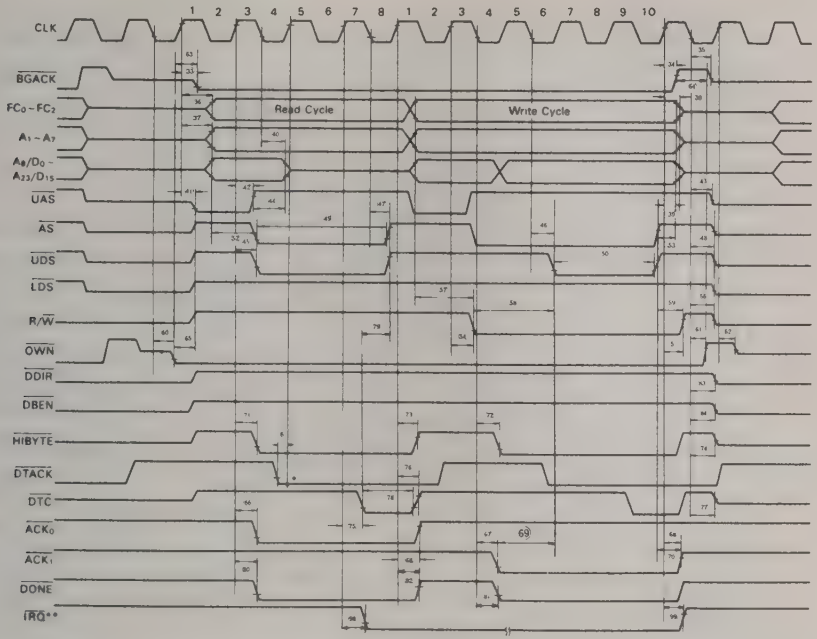
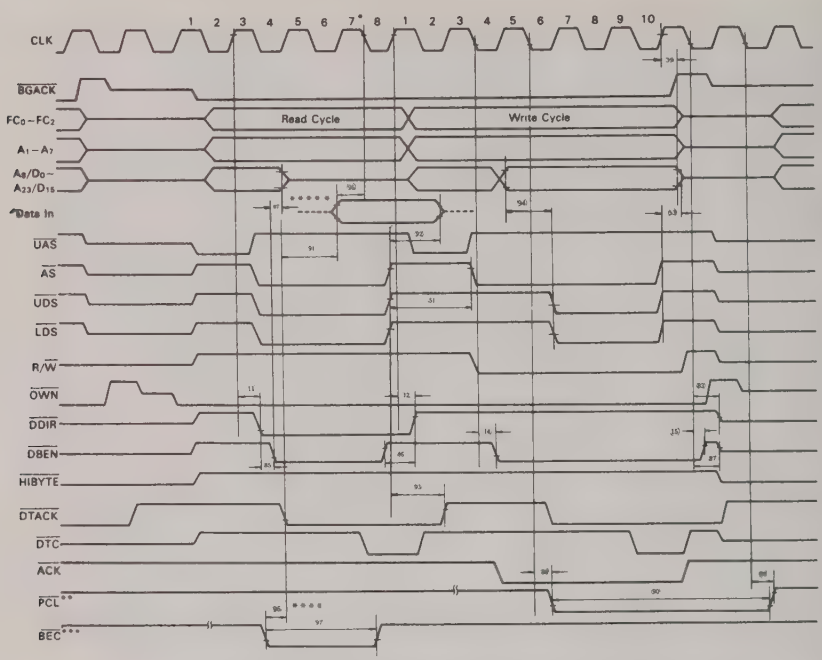


Figure 4 AC Electrical Waveforms — Bus Arbitration



* $\overline{\text{DTACK}}$ is picked up at the rising edge of CLK. This is different from HD68000.
** This timing is not related to DMA Read/Write (Single Cycle) sequence.

Figure 5 AC Electrical Waveforms – DMA Read/Write (Single Cycle)



* Data are latched at the end of clock 7. This timing is the same as HD68000.
** This timing is not related to DMA Read/Write (Dual Cycle) sequence. This timing is only applicable when 1/8 clock pulse mode is selected.
*** This timing is applicable when a bus exception occures.
**** If #6 is satisfied for both $\overline{DTACK}$ and $\overline{BEC}$, #96 may be Ons.
***** If the propagation delay of the external bidirectional buffer LS245 is less than 17nsec, the conflict may occur between the address output of the DMAC and the system data bus. In this case, the output of $\overline{DBEN}$ must be delayed externally.

Figure 6 AC Electrical Waveforms — DMA Read/Write (Dual Cycle)

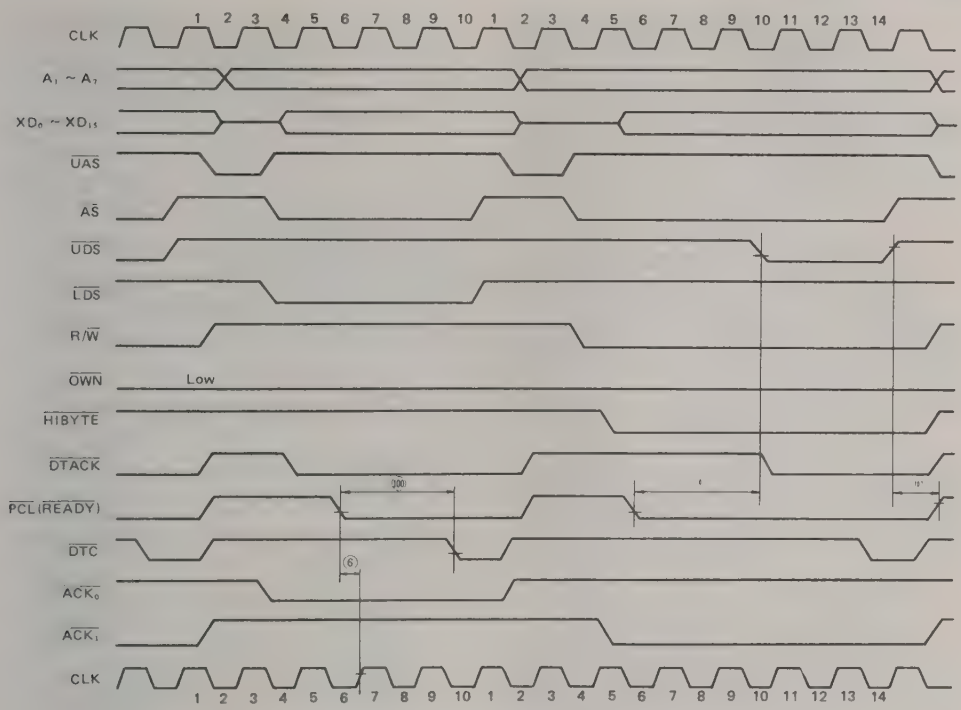
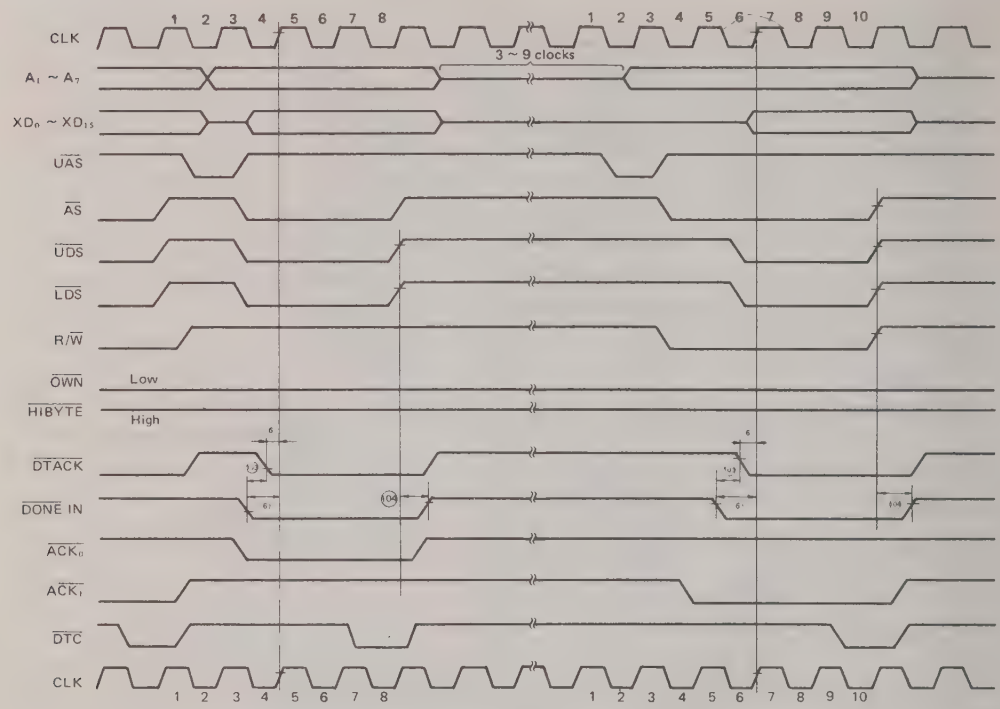


Figure 7 AC Electrical Waveforms — DMA Read/Write (Single Cycle with $\overline{PCL}$)



* If #6 is satisfied for both $\overline{DTACK}$ and $\overline{DONE}$, #103 may be 0ns.

Figure 8 AC Electrical Waveforms — $\overline{DONE}$ Input

(NOTES for Figure 3 through 8)

1) Setup time for the asynchronous inputs $\overline{BG}$, $\overline{BGACK}$, $\overline{CS}$, $\overline{IACK}$, $\overline{AS}$, $\overline{UDS}$, $\overline{LDS}$, and R/$\overline{W}$ guarantees their recognition at the next falling edge of the clock. Setup time for $\overline{BEC_0} \sim \overline{BEC_2}$, $\overline{REQ_0} \sim \overline{REQ_3}$, $\overline{PCL_0} \sim \overline{PCL_3}$, $\overline{DTACK}$, and $\overline{DONE}$ guarantees their recognition at the next rising edge of the clock.
2) Timing measurements are referenced to and from a low voltage of 0.8 volts and a high voltage of 2.0 volts.
3) These waveforms should only be referenced in regard to the edge-to-edge measurement of the timing specifications. They are not intended as a functional description of the input and output signals. Refer to other functional descriptions and their related diagrams for device operation.

■ SIGNAL DESCRIPTION

The following section identifies the signals used in the DMAC. In the definitions, "MPU mode" refers to the state when the DMAC is chip selected by MPU. The term "DMA mode" refers to the state when the DMAC assumes ownership of the bus. The DMAC is in the "IDLE mode" at all other times. Moreover, the DMA bus cycle refers to the bus cycle that is executed by the DMAC in the "DMA mode".

NOTE) In this data sheet, the state of the signals is described with these words: active or assert, inactive or negate.

This is done to avoid confusion when dealing with a mixture of "active-low" and "active-high" signals. The term assert or assertion is used to indicate that a signal is active or true independent of whether that voltage is low or high. The term negate or negation is used to indicate that a signal is inactive or false.

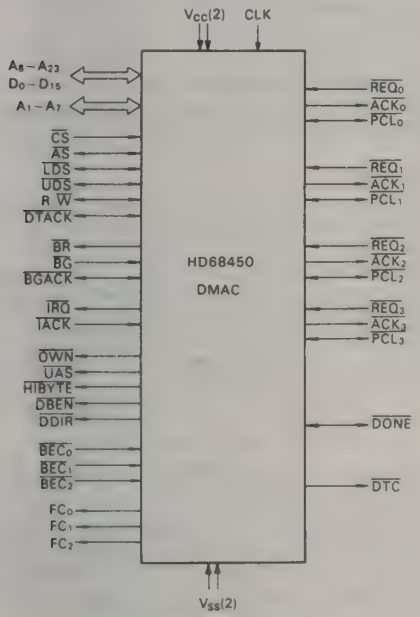


Figure 9 Input and Output Signals

- **Address/Data Bus (A_8/D_0 through A_{23}/D_{15})**

Input/Output Active-high	Three-statable

These lines are time multiplexed for address and data bus. The lines $\overline{DDIR}$, $\overline{DBEN}$, $\overline{UAS}$ and $\overline{OWN}$ are used to control the demultiplexing of the data and address lines externally. Demultiplexing is explained in the later section. The bi-directional data bus is used to transfer data between DMAC, MPU, memory and I/O devices.

Address lines are outputs to address memory and I/O devices.

- **Address Bus (A_1 through A_7)**

Input/Output Active-high	Three-statable

In the MPU mode, the DMAC internal registers are accessed with these lines and $\overline{LDS}$, $\overline{UDS}$. The address map for these registers is shown in Table 1. During a DMA bus cycle, A_1-A_7 are outputs containing the low order address bits of the location being accessed.

- **Function Code (FC_0 through FC_2)**

Output Active-high	Three-statable

These output signals provide the function codes during DMA bus cycles. They are three-stated except in the DMA bus cycles. They are used to control the HMCS68000 memories.

- **Clock (CLK)**

Input

This is the input clock to the HD68450, and should never be terminated at any time. This clock can be different from the MPU clock since HD68450 operates completely asynchronously.

- **Chip Select ($\overline{CS}$)**

Input Active low

This input signal is used to chip select the DMAC in "MPU" mode. If the $\overline{CS}$ input is asserted during a bus cycle which is generated by the DMAC, the DMAC internally terminates the bus cycle and signals an address error. This function protects the DMAC from accessing its own register.

- **Address Strobe ($\overline{AS}$)**

Input/Output Active low	Three-statable

In the "MPU mode", this line is an input indicating valid address input, and during the DMA bus cycle it is an output indicating valid the address output from the DMAC on the address bus.

The DMAC monitors these input lines during bus arbitration to determine the completion of the bus cycle by the MPU or other bus masters.

- **Upper Address Strobe ($\overline{UAS}$)**

Output Active low	Three-statable

This line is an output to latch the upper address lines on the multiplexed data/address lines. It is three-stated except in the "DMA mode".

- **Own ($\overline{OWN}$)**

Output Active low	Three-statable

This line is asserted by the DMAC during DMA mode, and is used to control the output of the address line latch. This line may also be used to control the direction of bi-directional buffers when loads on $\overline{AS}$, $\overline{LDS}$, $\overline{UDS}$, R/$\overline{W}$ and other signals exceed the drive capability. It is three-stated in the "MPU mode" and the "IDLE mode"

- **Data Direction ($\overline{DDIR}$)**

Outputs	Three-statable
Active low	(when data direction is input to the DMAC)
Active high	(when the data direction is output from the DMAC)

This line controls the direction of data through the bidirectional buffer which used to demultiplex the data/address lines. It is three-stated during the "IDLE mode"

- **Data Bus Enable ($\overline{DBEN}$)**

Output	Three-statable
Active low	

This line controls the output enable line of bidirectional buffers on the multiplexed data/address lines. It is a three-stated during the "IDLE mode".

- **High Byte (HIBYTE)**

Output	Three-statable
Active low	

This line is used when the operand size is byte in the single addressing mode. It is asserted when data is present on the upper eight bits of the data bus. It is used to control the output of bidirectional buffers which connects the upper eight bits of the data bus with the lower eight bits. It is three-stated during the "MPU mode" and the "IDLE mode"

- **Read/Write (R/$\overline{W}$)**

Input/Output	Three-statable
Active low (write)	
Active high (read)	

This line is an input in the "MPU mode" and an output during the "DMA mode". It is three-stated during the "IDLE mode". It is used to control the direction of data flow.

- **Upper Data Strobe ($\overline{UDS}$), Lower Data Strobe ($\overline{LDS}$)**

Input/Output	Three-statable
Active low	

These lines are extensions of the address lines indicating which byte or bytes of data of the addressed word are being addressed. These lines combined corresponds to address line A_0 in table 1.

- **Data Transfer Acknowledge ($\overline{DTACK}$)**

Input/Output	Three-statable
Active low	

In the "MPU mode", this line is an output indicating the completion of Read/Write bus cycle by the MPU.

In the "DMA mode", the DMAC monitors this line to determine when a data transfer has completed. In the event that a bus exception is requested, except for $\overline{HALT}$, prior to or concurrent with $\overline{DTACK}$, the $\overline{DTACK}$ response is ignored and the bus exception is honored. In the "IDLE mode", this signal is three-stated.

- **Bus Exception Controls ($\overline{BEC_0}$ through $\overline{BEC_2}$)**

Input
Active low

These lines provide an encoded signal input indicating an exceptional condition in the DMA bus cycle. See bus exception section for details.

- **Bus Request ($\overline{BR}$)**

Output
Active low

This output line is used to request ownership of the bus by the DMAC.

- **Bus Grant ($\overline{BG}$)**

Input
Active low

This line is used to indicate to the DMAC that it is to be the next bus master. The DMAC cannot assume bus ownership until both $\overline{AS}$ and $\overline{BGACK}$ becomes inactive. Once the DMAC acquires the bus, it does not continue to monitor the $\overline{BG}$ input.

- **Bus Grant Acknowledge ($\overline{BGACK}$)**

Input/Output	Three-statable
Active low	

Bus Grant Acknowledge ($\overline{BGACK}$) is a bidirectional control line. As an output, it is generated by the DMAC to indicate that it is the bus master.

As an input, $\overline{BGACK}$ is monitored by the DMAC, in limited rate auto-request mode, to determine whether or not the current bus master is a DMA device or not. $\overline{BGACK}$ is also monitored during bus arbitration in order to assume bus ownership.

- **Interrupt Request ($\overline{IRQ}$)**

Output	Open drain
Active low	

This line is used to request an interrupt to the MPU

- **Interrupt Acknowledge ($\overline{IACK}$)**

Input
Active low

This line is an input to the DMAC indicating that the current bus cycle is an interrupt acknowledge cycle by the MPU. The

DMAC responds the interrupt vector of the channel with the highest priority requesting an interrupt. There are two kinds of the interrupt vectors for each channel: normal (NIV) or error (EIV). $\overline{IACK}$ is not serviced if the DMAC has not generated $\overline{IRQ}$.

- **Channel Request ($\overline{REQ_0}$ through $\overline{REQ_3}$)**

Input	
Active low or falling edge	

These lines are the DMA transfer request inputs from the peripheral devices.

These lines are falling edge sensitive inputs when the request mode is cycle steal. They are low-level sensitive when the request mode is burst.

- **Channel Acknowledge ($\overline{ACK_0}$ through $\overline{ACK_3}$)**

Output	
Active low	

These lines indicate to the I/O device requesting a transfer that the request is acknowledged and the transfer is to be performed. These lines may be used as a part of the enable circuit for bus interface to the peripheral.

- **Peripheral Control Line ($\overline{PCL_0}$ through $\overline{PCL_3}$)**

Input/Output	Three-statable
Active low	

The four lines ($\overline{PCL_0} \sim \overline{PCL_3}$) are multi-purpose lines which may be individually programmed to be a START output, an Enable Clock input, a READY input, an ABORT input, a STATUS input, or an INTERRUPT input.

- **Done ($\overline{DONE}$)**

Input/Output	Open Drain
Active low	

As an output, this line is asserted concurrently with the $\overline{ACK_X}$ timing to indicate the last data transfer to the peripheral device. As an input, it allows the peripheral device to request a normal termination of the DMA transfer.

- **Device Transfer Complete ($\overline{DTC}$)**

Output	Three-statable
Active low	

This line is asserted when the DMA bus cycle has terminated normally with no exceptions. It may be used to supply the data latch timing to the peripheral device. In this case, data is valid at the falling edge of $\overline{DTC}$.

■ INTERNAL ORGANIZATION

The DMAC has four independent DMA channels. Each channel has its own set of channel registers. These registers define and control the activity of the DMAC in processing a channel operation.

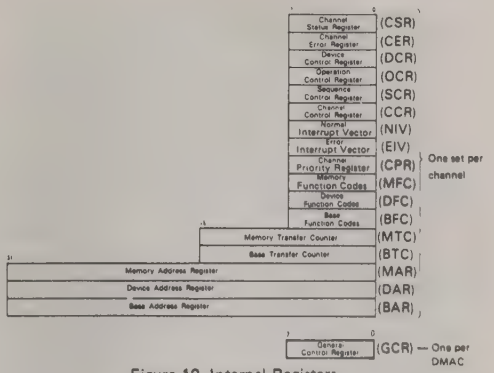

Channel Status Register (CSR)
Channel Error Register (CER)
Device Control Register (DCR)
Operation Control Register (OCR)
Sequence Control Register (SCR)
Channel Control Register (CCR)
Normal Interrupt Vector (NIV)
Error Interrupt Vector (EIV)
Channel Priority Register (CPR)
Memory Function Codes (MFC)
Device Function Codes (DFC)
Base Function Codes (BFC)
One set per channel
Memory Transfer Counter (MTC)
Base Transfer Counter (BTC)
Memory Address Register (MAR)
Device Address Register (DAR)
Base Address Register (BAR)
General Control Register (GCR)
One per DMAC

Figure 10 Internal Registers

- **Register Organization**

The internal register addresses are represented in Table 1. Address space not used within the address map is reserved for future expansion. A read from an unused location in the map results in a normal bus cycle with all ones for data. A write to one of these locations results in a normal bus cycle but no write occurs.

Unused bits of the defined registers in Table 1 read as zeros.

Table 1 Internal Register Addressing Assignments

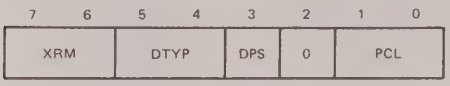
	Address Bits								
Register	7	6	5	4	3	2	1	0	Mode
Channel Status Register	c	c	0	0	0	0	0	0	R/W*
Channel Error Register	c	c	0	0	0	0	0	1	R
Device Control Register	c	c	0	0	0	0	1	0	R/W
Operation Control Register	c	c	0	0	0	1	0	1	R/W
Sequence Control Register	c	c	0	0	0	1	1	0	R/W
Channel Control Register	c	c	0	0	0	1	1	1	R/W
Memory Transfer Counter	c	c	0	0	1	0	1	b	R/W
Memory Address Register	c	c	0	0	1	1	s	s	R/W
Device Address Register	c	c	0	1	0	1	s	s	R/W
Base Transfer Counter	c	c	0	1	1	0	1	b	R/W
Base Address Register	c	c	0	1	1	1	s	s	R/W
Normal Interrupt Vector	c	c	1	0	0	1	0	1	R/W
Error Interrupt Vector	c	c	1	0	0	1	1	1	R/W
Channel Priority Register	c	c	1	0	1	1	0	1	R/W
Memory Function Codes	c	c	1	0	1	0	0	1	R/W
Device Function Codes	c	c	1	1	0	0	0	1	R/W
Base Function Codes	c	c	1	1	1	0	0	1	R/W
General Control Register	1	1	1	1	1	1	1	1	R/W

cc 00-Channel #0, 01-Channel #1,
10-Channel #2, 11-Channel #3.
ss 00-high-order, 01-upper middle,
10-lower middle, 11-low-order
b 0-high-order, 1-low-order
* see Channel Status Register Section

- **Device Control Register (DCR)**

The DCR is a device oriented control register. The XRM bits specifies whether the channel is in burst or cycle steal request mode. The DTYP bits define what type of device is on the channel. If the DTYP bits are programmed to be a HMCS6800 device, the PCL definition is ignored and the $\overline{PCL}$ line is an Enable clock input. If the DTYP bits are programmed to be a device with $\overline{READY}$, the PCL definition is ignored and the $\overline{PCL}$ line is a $\overline{READY}$ input. The DPS bit defines the port size (eight or sixteen bits) of the peripheral device. (A port size is the largest data which the peripheral device can transfer during a DMA bus cycle.) The PCL bits define the function of the $\overline{PCL}$ line. If the DTYP bits are programmed to be HMCS6800 device, or Device with $\overline{ACK}$ and $\overline{READY}$, these definitions are ignored. The XRM

bits are ignored if an auto-request mode (REQG = 00 or 01 in Operation Control Register) is selected.



XRM (EXTERNAL REQUEST MODE)
- 00 Burst Transfer Mode
- 01 (undefined, reserved)
- 10 Cycle Steal Mode without Hold
- 11 Cycle Steal Mode with Hold

DTYP (DEVICE TYPE)
- 00 HD68000 compatible device, explicitly addressed (dual addressing mode)
- 01 HD6800 compatible device, explicitly addressed (dual addressing mode)
- 10 Device with $\overline{ACK}$, implicitly addressed (single addressing mode)
- 11 Device with $\overline{ACK}$ and $\overline{READY}$, implicitly addressed (single addressing mode)

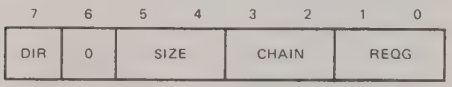
DPS (DEVICE PORT SIZE)
- 0 8 bit port
- 1 16 bit port

PCL (PERIPHERAL CONTROL LINE)
- 00 Status Input
- 01 Status Input with Interrupt
- 10 Start Pulse
- 11 Abort Input

Bit 2 Not Used

- **Operation Control Register (OCR)**

The OCR is an operation control register. The DIR bit defines the direction of the transfer. The SIZE bits define the size of the operand. The CHAIN bits define the type of the CHAIN mode. The REQG bits define how requests for transfers are generated.



DIR (DIRECTION)
- 0 Transfer from memory to device (transfer from MAR address to DAR address)
- 1 Transfer from device to memory (transfer from DAR address to MAR address)

SIZE (OPERAND SIZE)
- 00 Byte (8 bits)
- 01 Word (16 bits)
- 10 Long Word (32 bits)
- 11 (undefined, reserved)

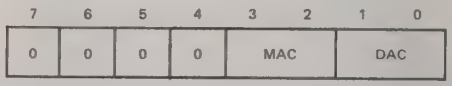
CHAIN (CHAINING OPERATION)
- 00 Chain operation is disabled
- 01 (undefined, reserved)
- 10 Array Chaining
- 11 Linked Array Chaining

REQG (DMA REQUEST GENERATION METHOD)
- 00 Auto-request at transfer rate limited by General Control Register (Limited Rate Auto-Request)
- 01 Auto-request at maximum rate
- 10 $\overline{REQ}$ line requests an operand transfer
- 11 Auto-request the first operand, external request for subsequent operands

Bit 6 Not Used

- **Sequence Control Register (SCR)**

The SCR is used to define the sequencing of memory and device addresses.



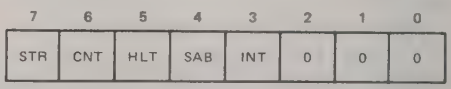
MAC (MEMORY ADDRESS COUNT)
- 00 Memory address register does not count
- 01 Memory address register counts up
- 10 Memory address register counts down
- 11 (undefined, reserved)

DAC (DEVICE ADDRESS COUNT)
- 00 Device address register does not count
- 01 Device address register counts up
- 10 Device address register counts down
- 11 (undefined, reserved)

Bits 7, 6, 5, 4 Not Used

- **Channel Control Register (CCR)**

The CCR is used to start or terminate the operation of a channel. This register also determines if an interrupt request is to be generated. Setting the STR bit causes immediate activation of the channel; the channel will be ready to accept request immediately. The STR and CNT bits of the register cannot be reset by a write to the register. The SAB bit is used to terminate the operation forcedly. Setting the SAB bit will reset STR and CNT. Setting the HLT bit will halt the channel operation, and clearing the HLT bit will resume the operation. Setting start bit must be done by byte access. Otherwise, timing error occurs.



STR (START OPERATION)
- 0 No operation is pending
- 1 Start operation

CNT (CONTINUE OPERATION)
- 0 No continuation is pending
- 1 Continue operation

HLT (HALT OPERATION)
- 0 Operation not halted
- 1 Operation halted

SAB (SOFTWARE ABORT)
- 0 Channel operation not aborted
- 1 Abort channel operation

INT (INTERRUPT ENABLE)
- 0 No interrupts enabled
- 1 Interrupts enabled

Bits 2, 1, 0 Not Used

- **Channel Status Register (CSR)**

The CSR is a register containing the status of the channel.

7	6	5	4	3	2	1	0
COC	BTC	NDT	ERR	ACT	0	PCT	PCS

COC (CHANNEL OPERATION COMPLETE)
0 Channel operation incomplete
1 Channel operation complete
BTC (BLOCK TRANSFER COMPLETE)
0 Block transfer incomplete
1 Block transfer complete
NDT (NORMAL DEVICE TERMINATION)
0 No normal device termination by $\overline{\text{DONE}}$ input
1 Device terminated operation normally by $\overline{\text{DONE}}$ input
ERR (ERROR BIT)
0 No errors
1 Error as coded in CER
ACT (CHANNEL ACTIVE)
0 Channel not active
1 Channel active
PCT ($\overline{\text{PCL}}$ TRANSITION)
0 No $\overline{\text{PCL}}$ transition occurred
1 $\overline{\text{PCL}}$ transition occurred
PCS (THE STATE OF THE $\overline{\text{PCL}}$ INPUT LINE)
0 $\overline{\text{PCL}}$ "Low"
1 $\overline{\text{PCL}}$ "High"
Bit 2 Not Used

- **Channel Error Register (CER)**

The CER is an error condition status register. The ERR bit of CSR indicates if there is an error or not. Bits 0-4 indicate what type of error occurred.

7	6	5	4	3	2	1	0
0	0	0	ERROR CODE				

Error Code
00000 No error
00001 Configuration error
00010 Operation timing error
00101 Address error in MAR
00110 Address error in DAR
00111 Address error in BAR
01001 Bus error in MAR
01010 Bus error in DAR
01011 Bus error in BAR
01101 Count error in MTC
01111 Count error in BTC
10000 External abort
10001 Software abort
Bits 7, 6, 5 Not Used

- **Channel Priority Register (CPR)**

The CPR is used to define the priority level of the channel. Priority level 0 is the highest and priority level 3 is the lowest priority.

7	6	5	4	3	2	1	0
0	0	0	0	0	0	CP	

CP (CHANNEL PRIORITY)
00 Priority level 0
01 Priority level 1
10 Priority level 2
11 Priority level 3
Bit 7 through 2 Not Used

- **General Control Register (GCR)**

The GCR is used to define what portion of the bus cycles is available to the DMAC for limited rate auto-request generation. GCR is also used to specify the hold time for cycle steal mode with hold.

7	6	5	4	3	2	1	0
0	0	0	0	BT		BR	

BT (BURST TIME)

The number of DMA clock cycles per burst that the DMAC allows in the auto-request at a limited rate of transfer is controlled by these two bits. The number is 2(BT+4) (two to the BT+4 power).

BR (BANDWIDTH RATIO)

The amount of the bandwidth utilized by the auto-request at a limited rate transfer is controlled by these two bits. The ratio is 2(BR+1) (two to the BR+1 power).

The hold time for cycle steal mode with hold is defined to be minimum of 1 sample interval and maximum of 2 sample intervals. A sample interval is defined to be 2(BT+BR+5) (two to the BT+BR+5 power) clock cycles.

Bits 7 through 4 Not Used

- **Address Registers (MAR, DAR, BAR)**

Three 32-bit registers are utilized to implement the Memory Address Register, Device Address Register, and the Base Address Register. Only the least significant twenty-four bits are connected to the address output pins. The content of the MAR is outputted when the memory is accessed in single or dual adressing mode. The content of the DAR is outputted when the peripheral device is accessed. The contents of the BAR is outputted when reading chain information from memory in the Array Chaining Mode or the Linked Array Chaining Mode. It is also used to set the top address of the next block transfer in Continue mode.

- **Function Code Registers (MFC, DFC, BFC)**

The DMAC has three function code registers per channel: the Memory Function Code Register (MFC), Device Function Code Register (DFC), and the Base Function Code Register (BFC). The contents of these registers are outputted from FC_0 through FC_2 lines when an address is outputted from MAR, DAR, or BAR, respectively. The BFC is also used to set the MFC for the transfer of the next data block in the Continue mode.

7	6	5	4	3	2	1	0
0	0	0	0	0	FC2	FC1	FC0

Bits 3 through 7 Not Used

- **Transfer Count Registers (MTC, BTC)**

Each channel has two 16-bit counters: the Memory Transfer Counter (MTC) and the Base Transfer Counter (BTC). The MTC

counts the number of transfer words in one block, and is decreased by one for every operand transfer.

The BTC is used to count the number of data blocks in the Array Chaining Mode. BTC is also used to set the number of operands to transfer for the next data block in the Continue Mode.

- **Interrupt Vector Registers (NIV, EIV)**

Each channel has a Normal Interrupt Vector register and an Error Interrupt Vector register.

When an interrupt acknowledge cycle occurs, an interrupt vector is outputted from one of those registers. If the error bit (CSR) is set for the channel with interrupt pending, then content of EIV is outputted, otherwise content of NIV is outputted.

■ OPERATION DESCRIPTION

A DMAC channel operation proceeds in three principal phases. During the initialization phase, the MPU sets the channel control registers, supply the initial address and the number of transfer words, and starts the channel. During the transfer phase, the DMAC accepts requests for data operand transfers, and provides addressing and bus controls for the transfers. The termination phase occurs after the operation is completed.

This section describes DMAC operations. A description of the MPU/DMAC communication is given first. Next, the transfer phase is covered, including how the DMAC recognizes requests and how the DMAC arranges for data transfer. Following this, the initialization phase is described. The termination phase is covered, introducing chaining, error signaling, and bus exceptions. A description of the channel priority scheme rounds out the section.

- **Read/Write of the DMAC Registers by MPU**

The MPU reads and writes the DMAC internal registers and controls the DMA transfer. Figure 11 indicates the timing diagram when the MPU reads the contents of the DMAC register. The MPU outputs A_1-A_{23}, FC_0-FC_2, $\overline{AS}$, R/$\overline{W}$, $\overline{UDS}$, and $\overline{LDS}$, and accesses the DMAC internal register. The specific internal register is selected by A_1-A_7, $\overline{LDS}$ and $\overline{UDS}$. The $\overline{CS}$ and $\overline{IACK}$ lines are generated by the external circuit with A_8-A_{23} and FC_0-FC_2. The DMAC outputs data on the data bus, together with $\overline{DDIR}$, $\overline{DBEN}$ and $\overline{DTACK}$. The $\overline{DDIR}$ and $\overline{DBEN}$ control the bidirectional buffer on the bus and the $\overline{DTACK}$ indicates that the data has been sent or received by the DMAC. Read Cycle is eighteen CLKs. Figure 12 shows the MPU write cycle. Write cycle is fifteen CLKs.

Note the following points.

(1) The clock reference shown in this figure is the DMAC input clock.
(2) The $\overline{DDIR}$ and the $\overline{DBEN}$ are three-stated at the beginning which detects $\overline{CS}$ and the ending of the cycle.
(3) During the MPU read cycle, the $\overline{DTACK}$ is asserted after the data is valid on the system bus.
(4) During the MPU write cycle, the $\overline{DDIR}$ line will be driven low to direct the data buffers toward to DMAC before the buffers are enabled.
(5) During the MPU write cycle, the DMAC will latch the data before asserting $\overline{DTACK}$. Then it will negate $\overline{DBEN}$ and $\overline{DDIR}$ in the proper order.
(6) After the MPU cycle and the $\overline{LDS}$ and the $\overline{UDS}$ are negated by the MPU, the DMAC will put $\overline{DBEN}$, $\overline{DDIR}$ and the address data lines to a high impedance state.
(7) $\overline{DTACK}$ will once go "High" and then to a high impedance state after negating $\overline{LDS}$ and $\overline{UDS}$.

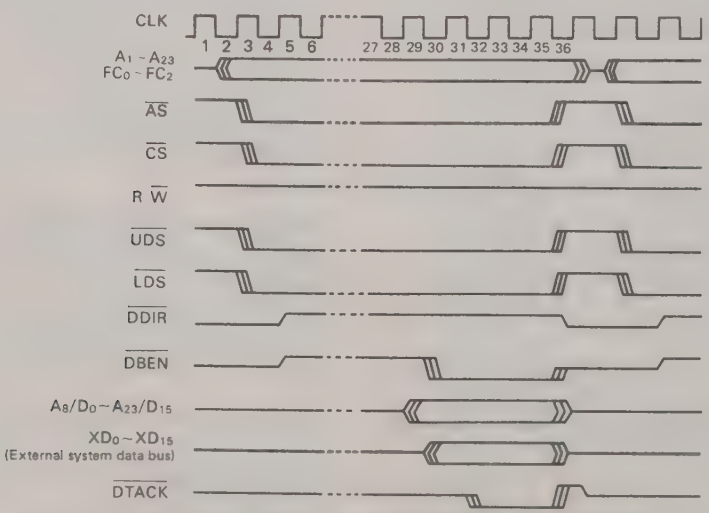


Figure 11 MPU Read from DMAC – Word

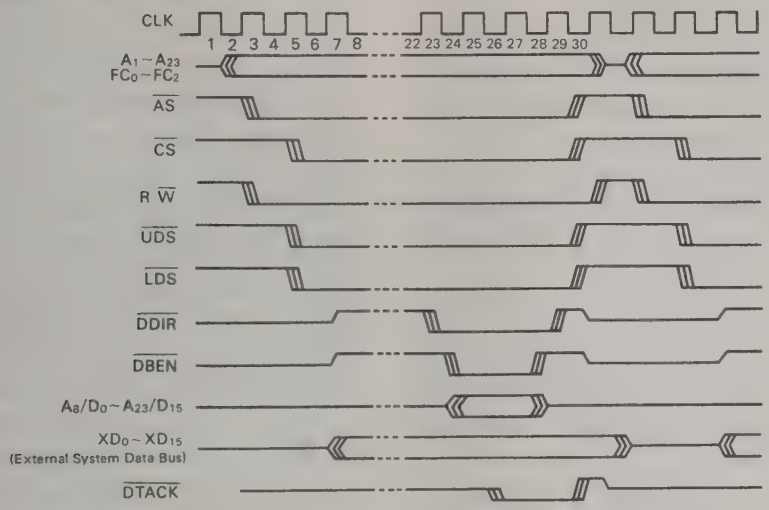
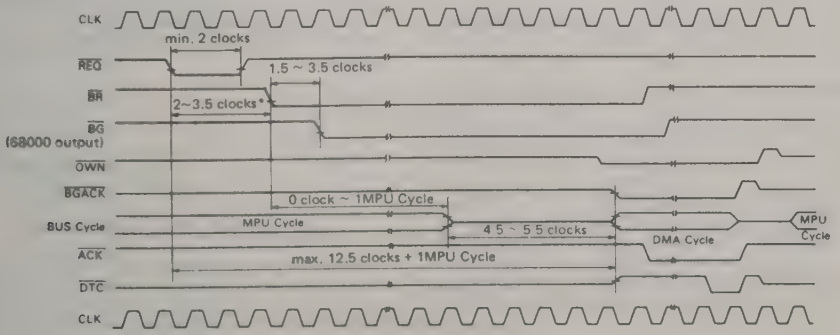


Figure 12 MPU Write to DMAC – Word

- **Bus Arbitration**

The followings are the description of the bus arbitration. The DMAC must obtain the ownership of the bus in order to transfer data. Figure 13 indicates the DMAC bus arbitration timing. It is completely compatible with that of HD68000 MPU. The DMAC asserts the Bus Grant ($\overline{BG}$) to request the bus mastership. The MPU recognizes the request and asserts $\overline{BG}$, then it grants the ownership in the next bus cycle. After the end of the current cycle ($\overline{AS}$ is negated), the MPU relinquishes the bus to the DMAC. The DMAC asserts the bus grant acknowledge ($\overline{BGACK}$) to indicate that it has the bus ownership. A half clock before $\overline{BGACK}$ is asserted, the DMAC asserts $\overline{OWN}$. $\overline{OWN}$ is kept asserted for a half clock after $\overline{BGACK}$ is negated at the end of the DMA cycle. $\overline{BR}$ is negated one clock after $\overline{BGACK}$ is asserted.



* This case assumes that no exception condition exists and DMAC isn't accessed by MPU.

Figure 13 DMAC Bus Arbitration Timing

- **Device/DMAC Communication**

Communication between peripheral devices and the DMAC is accomodated by five signal lines. Each channel has $\overline{REQ}$, $\overline{ACK}$ and $\overline{PCL}$, and the last two lines the $\overline{DONE}$ and $\overline{DTC}$ lines, are shared among the four channels.

(1) Request ($\overline{REQ}$)

The peripheral devices assert $\overline{REQ}$ to request data transfers. See the "Requests" section for details.

(2) Acknowledge ($\overline{ACK}$)

This line is used to implicitly address the device which is transferring the data (This device is not selected by address lines.) It is also asserted when the content of DAR is outputted during memory-to-memory transfer except for the auto-request mode at a limited rate or at the maximum rate.

(3) Peripheral Control Line ($\overline{PCL}$)

The function of this line is quite flexible and is determined by the DCR (Device Control Register).

The DTYP bits of the DCR define what type of device is on the channel. If the DTYP bits are programmed to be a HMCS6800 device, the PCL definition is ignored and the $\overline{PCL}$ line is an Enable clock (E clock) input. If the DTYP bits are programmed to be a device with $\overline{READY}$, the PCL definition as ignored and the $\overline{PCL}$ line is a ready input.

$\overline{PCL}$ As a Status Input

The $\overline{PCL}$ line may be programmed as a status input. The status level of this line can be determined by the PCS bit in the CSR, regardless of the PCL function determined by the DCR. If a negative transition occurs and remains stable for a minimum of two clocks, the PCT bit of the CSR is set. This PCT bit is cleared by resetting the DMAC or the writing "1" to the PCT bit.

$\overline{PCL}$ As an Interrupt

The $\overline{PCL}$ line may be programmed to generate an interrupt on a negative transition. This enables an interrupt which is requested if the PCT bit of the CSR is set. When using this function, it is necessary to reset the PCT bit in the CSR before the PCL bit in the DCR is set to interrupt, in order to avoid assertion of IRQ line at this time.

$\overline{PCL}$ As a Starting Pulse

The $\overline{PCL}$ line may be programmed to output a starting pulse. This active low starting pulse is outputted when a channel is activated, and is "Low" for a period of four clock cycles.

$\overline{PCL}$ As an Abort Input

The $\overline{PCL}$ line may be programmed to be a negative transition above input which terminates an operation by setting the external abort error in CER. It is necessary to reset the PCT bit in the CSR before activating the channel (Setting the ACT bit of CCR) so that the channel operation is not immediately aborted.

$\overline{PCL}$ As an Enable Clock (E Clock) Input

If the DTYP bits are programmed to be a HMCS6800 device, the PCL definition is ignored and the $\overline{PCL}$ line is an Enable Clock input. The Enable clock downtime must be as long as five clock cycles, and must be high for a minimum of three DMAC clock cycles, but need not be synchronous with the DMAC's clock.

$\overline{PCL}$ As a $\overline{READY}$ Input

If the DTYP bits are programmed to be a device with $\overline{READY}$, the PCL definition is ignored and the $\overline{PCL}$ line is a $\overline{READY}$ input. The $\overline{READY}$ is an active low input.

(4) DONE ($\overline{DONE}$)

This line is an active low Input/Output signal with an open drain. It is asserted when the memory transfer count is exhausted in a single block transfer. In the chaining operation, $\overline{DONE}$ is asserted only at the last transfer to the peripheral device of the last data block. In the continue mode, $\overline{DONE}$ is asserted for each data block. It is asserted and negated in coincident with the $\overline{ACK}$ line for the last data transfer to the peripheral device. It is also outputted in coincident with the $\overline{ACK}$ line of the last bus cycle, in which the address is outputted from the DAR, in the memory-to-memory transfer (dual addressing mode) that uses the $\overline{ACK}$ line.

The DMAC also monitors the state of the $\overline{DONE}$ line during the DMA bus cycle. If the device asserts $\overline{DONE}$ during $\overline{ACK}$ active, the DMAC will terminate the operation after the transfer of the current operand. If $\overline{DONE}$ is asserted on the first byte of 2 byte operation or the first word of long word operation, the DMAC does not terminate the operation before the whole operand transfer is completed. If $\overline{DONE}$ is asserted, then the DMAC terminates the operation by clearing the ACT bit of the CSR, and setting the COC and NDT bits of the CSR. If both the DMAC and the device assert $\overline{DONE}$, the device termination is not recognized, but the channel operation does terminate. $\overline{DONE}$ is outputted again for the retry exceptions bus cycles.

(5) Data Transfer Complete ($\overline{DTC}$)

$\overline{DTC}$ is an active low signal which is asserted when the actual data transfer is accomplished. It is also asserted in the bus cycle which read a chain information from memory in the Chaining mode. However, if exceptions are generated and the DMA bus cycle terminates, $\overline{DTC}$ is not asserted. $\overline{DTC}$ is asserted one half clock before $\overline{LDS}$ and $\overline{UDS}$ are negated, and negated one half clock after $\overline{LDS}$ and $\overline{UDS}$ are negated.

- **Requests**

Requests may be externally generated by circuitry in the peripheral device, or internally generated by the auto-request mechanism. The REQG bits of the OCR determine these modes. The DMAC also supports an operation in which the DMAC auto-requests the first transfer and then wait for the peripheral device to request the following transfers.

(1) Auto-request Transfers

The auto-request mechanism provides generation of requests within the DMAC. These requests can be generated at either of two rates: maximum-rate and limited-rate. In the former case, the channel always has a request pending.

The limited rate auto-request functions by monitoring the bus utilization.

Limited-rate Auto-request

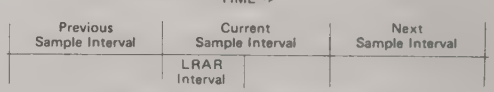


Figure 14 DMAC Sample Intervals

In the limited-rate auto-request the DMAC devides time into equal length sample intervals by counting clock cycles. The end of one sample interval makes the beginning of the next. During a sample interval, the DMAC monitors by means of $\overline{BGACK}$ pin the system bus activity of the DMAC and other bus master devices. At the end of the sample interval, decision is made whether or not to perform the channel's data transfer during the next sample interval. Namely, based on the activity of the DMAC or other bus master devices during the current sample interval, the DMAC allows limited-rate auto-requests for some initial portion of the next sample interval.

The length of the sample interval, and the portion of the sample interval during which limited-rate auto-requests can be

made (the limited-rate auto-request interval) are controlled by the BT and BR bits in the GCR. The length in clock cycles of the limited-rate auto-request interval is $2^{(BT+4)}$ (2 raised to the BT+4 power). For example, if BT equals 2 and the DMA utilization of the bus was low during the previous sample interval, then the DMAC generates the auto-request transfers during the first 64 clock cycles.

The ratio of the length of the sample interval to the length of the limited-rate auto-request interval is controlled by the BR bits. The ratio of the system bus utilization of the MPU to other bus master devices including he DMAC is $2^{(BR+1)}$ (2 raised to the BR+1 power). If the fraction of DMA clock cycles during the sample interval exceeds the programmed utilization level, the DMAC will not allow limited-rate auto-requests during the next sample interval.

For example, if BR equals 3, then at most one out of 16 clock cycles during a sample interval can be used by the DMAC and other bus master devices, and still the DMAC would allow limited rate auto-request during the next sample interval. Therefore, from the viewpoint of long period, the ratio of the system bus utilization of the MPU to I/O devices including the DMAC is about 16:1. The sample interval length is not a direct parameter, but it is equal to $2^{(BT+BR+5)}$ clock cycles. Thus, the sample interval can be programmed between 32 and 2048 clock cycles.

The DMAC uses the $\overline{BGACK}$ to differentiate between the MPU bus cycle and DMAC or other bus master devices. If $\overline{BGACK}$ is active, then the DMAC assumes that the bus is used by a DMAC or other bus master devices. If it is inactive, then the DMAC assumes that it is used by the MPU.

Maximum-rate Auto-request

If the REQG bits in the OCR indicate auto-request at the maximum rate, the DMAC acquires the bus after the start bit is set and keeps it until the data transfer is completed.

If a request is made by another channel of higher priority, the DMAC services that channel and then resumes the auto-request sequence. If two or more channels are set to equal priority level and maximum rate auto-request, then the channels will rotate in a "round robbin" fashion.

If the HMCS68000 compatible device is connected to a channel, the $\overline{ACK}$ line is held inactive during an auto-request operation. Consequently, any channel may be used for the memory-to-memory transfer with the auto-request function in addition to the operation of data transfer between memory and peripheral device with using the $\overline{REQ}$ pin. Refer to Figure 15 for the timing of the memory-to-memory transfer. In this mode, the $\overline{ACK}$, HIBYTE and $\overline{DONE}$ outputs are always inactive.

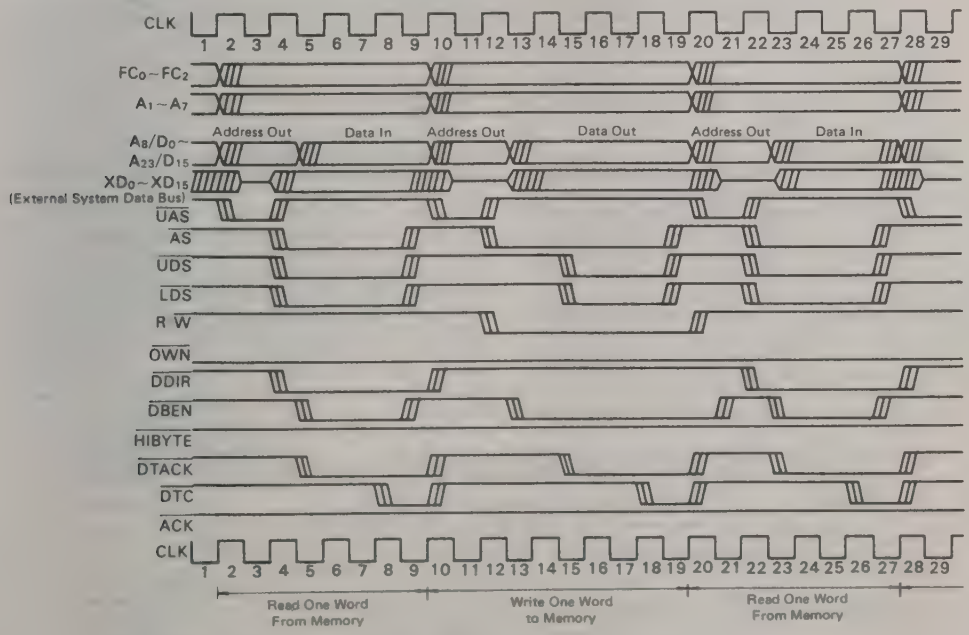


Figure 15 Memory-to-Memory Transfer Read-Write-Read Cycles

(2) External Requests

If the REQG bits of the OCR indicate that the $\overline{REQ}$ line generates requests, the transfer requests are generated externally. The request line associated with each channel allows the device to externally generate requests for DMA transfers. When the device wants an operand transferred, it makes a request by asserting the request line. The external request mode is determined by the XRM bits of the DCR, which allows both burst and cycle steal request modes. The burst request mode allows a channel to request the transfer of multiple operands using consecutive bus cycles. The cycle steal request mode allows a channel to request the transfer of a single operand. The followings are the description of the burst and the cycle steal modes.

Burst Request Recognition

In the burst request mode, the $\overline{REQ}$ line is an active low input. The level sampled at the rising edge of the clock. Once the burst request is asserted, it needs to be held low until the first DMA bus cycle starts in order to insure at least one data transfer operation. In order to stop the burst mode transfer after the current bus cycle, the $\overline{REQ}$ line has to be negated one clock before the $\overline{DTC}$ output clock of this cycle. Refer to Figure 16 or the burst mode timing.

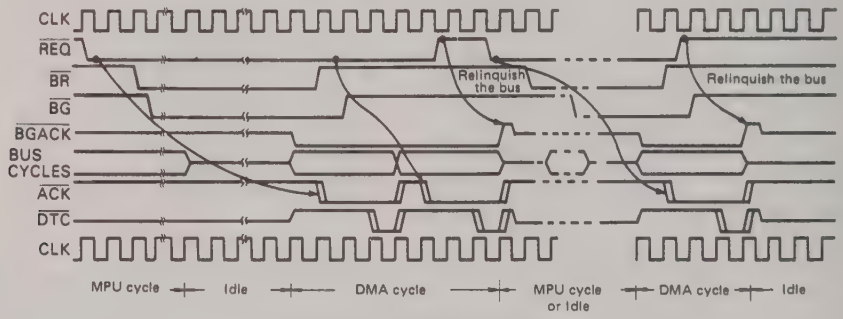


Figure 16 Burst Mode Request Timing
(Only one channel is active)

Cycle Steal Request Recognition

In the cycle steal request mode, the peripheral device requests the DMA transfer by generating an falling edge at the $\overline{REQ}$ line. The $\overline{REQ}$ line needs to be held "low" for at least 2 clock cycles. In the cycle steal mode, if the $\overline{REQ}$ line changes from "High" to "Low" between $\overline{ACK}$ output and one clock before the clock that outputs $\overline{DTC}$, then the next DMA transfer is performed without relinquishing the bus. If the bus is not relinquished, then maximum of 5 idle clocks is inserted between bus cycles. Refer to Figure 17 for the request timing of the cycle steal mode. If the XRM bits specify cycle steal without hold, the DMAC will relinquish the bus. If the XRM bits specify cycle steal with hold, the DMAC will retain ownership. The bus is not given up for arbitration until the channel operation terminates or until the device pauses. The device is determined to have paused if it does not make any requests during the next full sample interval. The sample interval counter is free running and is not reset or modified by this mode of operation. The sample interval counter is the same counter that is used for Limited Rate Auto Request and is programmed via the GCR. Figure 18 shows the request timing in the cycle steal bus hold. If the $\overline{REQ}$ is inputted during the hold time, the $\overline{ACK}$ is outputted after a maximum of 7.5 clock cycles from the picked-up clock. On the cycle steal with hold mode, the DMAC will hold the bus even when the transfer count is exhausted and the last data has been transferred. If DMA transfer is requested from other channels during this period, they are executed normally.

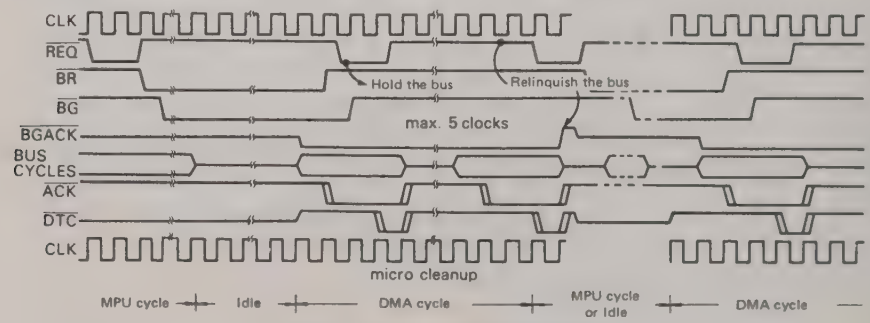


Figure 17 Cycle Steal Mode Request Timing

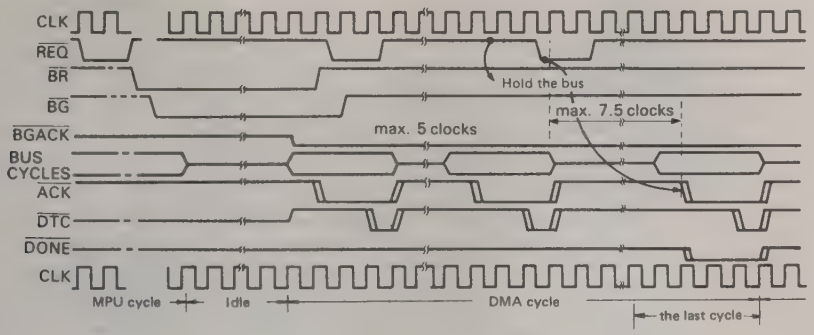


Figure 18 Cycle Steal Bus Hold Mode Request Timing

Request Recognition in Dual-address Transfers

In a following section dual-address transfers are defined. Dual address transfer is an exception to the request recognition rules in the previous paragraphs. In the cycle steal request mode, when there are two or more than transfers between the DMAC and the peripheral device during one operand transfer, the request is not recognized until the last transfer between the DMAC and the I/O device starts.

(3) Mixed Request Generation

A single channel can mix the two request generation methods. By programming the REQG bits of the OCR to "11", when the channel is started, the DMAC auto-requests the first transfer. Subsequent requests are then generated externally by the device. The $\overline{ACK}$ and $\overline{PCL}$ lines perform their normal functions in this operation.

- **Data Transfers**

All DMAC data transfers are assumed to be between memory and the peripheral device. The word "memory" means a 16-bit HMCS68000 bus compatible device. By programming the DCR, the characteristics of the peripheral device may be assigned. Each channel can communicate using any of the following protocols.

DTYP	Device Type	
00	HMCS68000 compatible device	Dual Addressing
01	HMCS6800 compatible device	
10	Device with $\overline{ACK}$	Single Addressing
11	Device with $\overline{ACK}$ and $\overline{READY}$	

(1) Dual Addressing

HMCS68000 and HMCS6800 compatible devices may be explicitly addressed. This means that before the peripheral transfers data, a data register within the device must be addressed. Because the address bus is used to address the peripheral, the data cannot be directly transferred to/from the memory because the memory also requires addressing. Instead, the data is transferred from the source to the DMAC and held in an internal DMAC holding register. A second bus transfer between the DMAC and the destination is then required to complete the operation. Because both the source and destination of the transfer are explicitly addressed, this protocol is called dual-addressed.

HMCS68000 Compatible Device Transfers

In this operation, when a request is received, the bus is obtained and the transfer is completed using the protocol as shown in Figures 19 and 20. Figures 21 through 24 show the transfer timings. Figure 21 and 24 show the operation when the memory is the source and the peripheral device is the destination. Figures 22 and 23 show the transfer in the opposite direction. The peripheral device is a 16-bit device in Figures 21 and 22, and a 8-bit device in Figures 23 and 24.

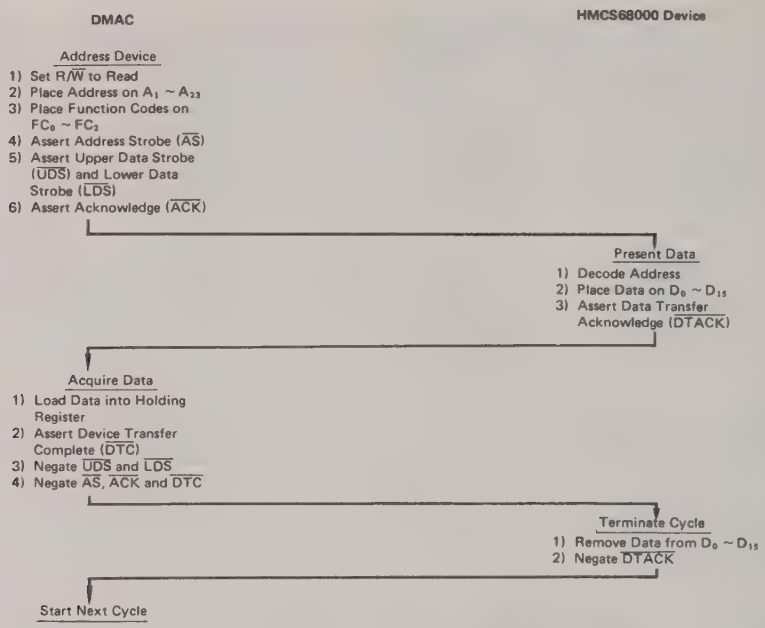

DMAC
HMCS68000 Device
Address Device
1) Set R/W to Read
2) Place Address on A1 ~ A23
3) Place Function Codes on FC0 ~ FC2
4) Assert Address Strobe (AS)
5) Assert Upper Data Strobe (UDS) and Lower Data Strobe (LDS)
6) Assert Acknowledge (ACK)
Present Data
1) Decode Address
2) Place Data on D0 ~ D15
3) Assert Data Transfer Acknowledge (DTACK)
Acquire Data
1) Load Data into Holding Register
2) Assert Device Transfer Complete (DTC)
3) Negate UDS and LDS
4) Negate AS, ACK and DTC
Terminate Cycle
1) Remove Data from D0 ~ D15
2) Negate DTACK
Start Next Cycle

Figure 19 Word Read Cycle Flowchart HMCS68000 Type Device

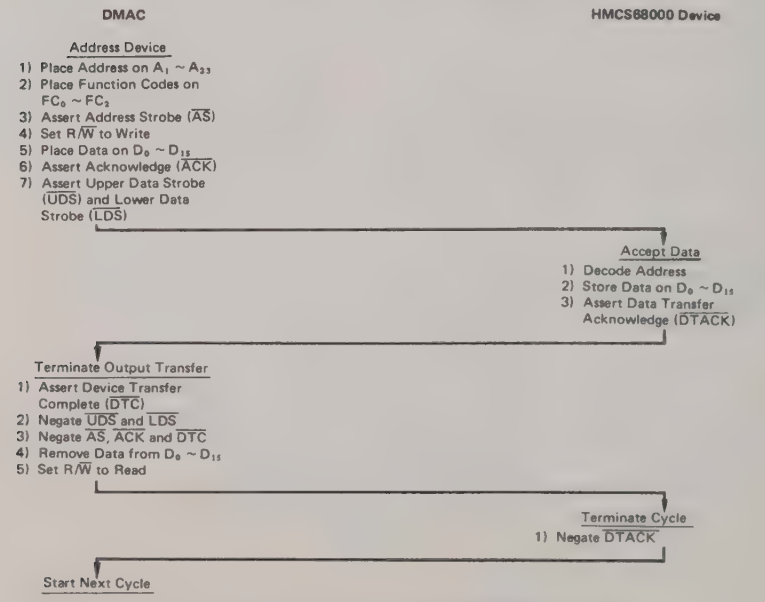

DMAC
HMCS68000 Device
Address Device
1) Place Address on A1 ~ A23
2) Place Function Codes on FC0 ~ FC2
3) Assert Address Strobe (AS)
4) Set R/W to Write
5) Place Data on D0 ~ D15
6) Assert Acknowledge (ACK)
7) Assert Upper Data Strobe (UDS) and Lower Data Strobe (LDS)
Accept Data
1) Decode Address
2) Store Data on D0 ~ D15
3) Assert Data Transfer Acknowledge (DTACK)
Terminate Output Transfer
1) Assert Device Transfer Complete (DTC)
2) Negate UDS and LDS
3) Negate AS, ACK and DTC
4) Remove Data from D0 ~ D15
5) Set R/W to Read
Terminate Cycle
1) Negate DTACK
Start Next Cycle

Figure 20 Word Write Cycle Flowchart HMCS68000 Type Device

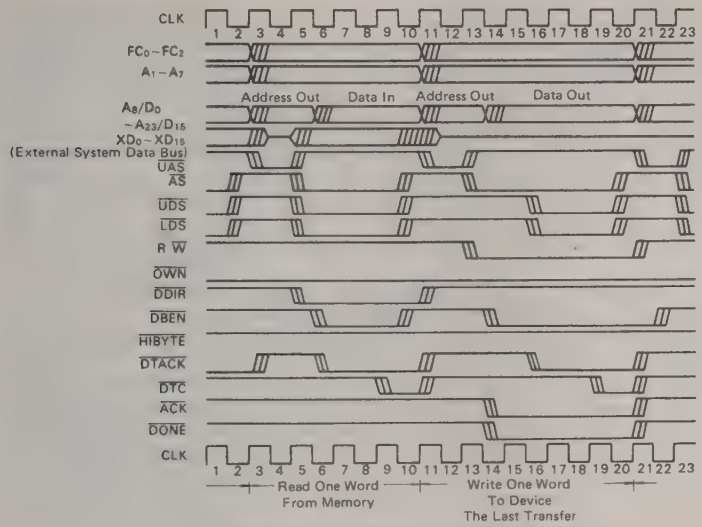


Figure 21 Dual Addressing Mode, Read/Write Cycle, Destination = 16-bit Device, Word Operand

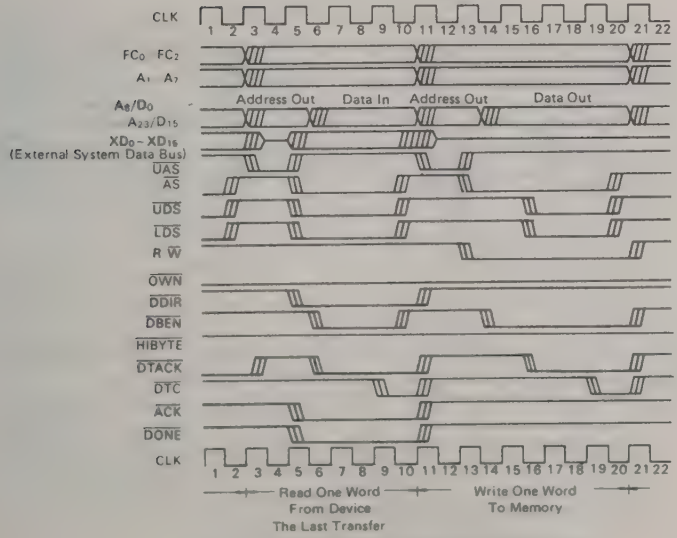


Figure 22 Dual Addressing Mode, Read/Write Cycle, Source = 16-bit Device, Word Operand

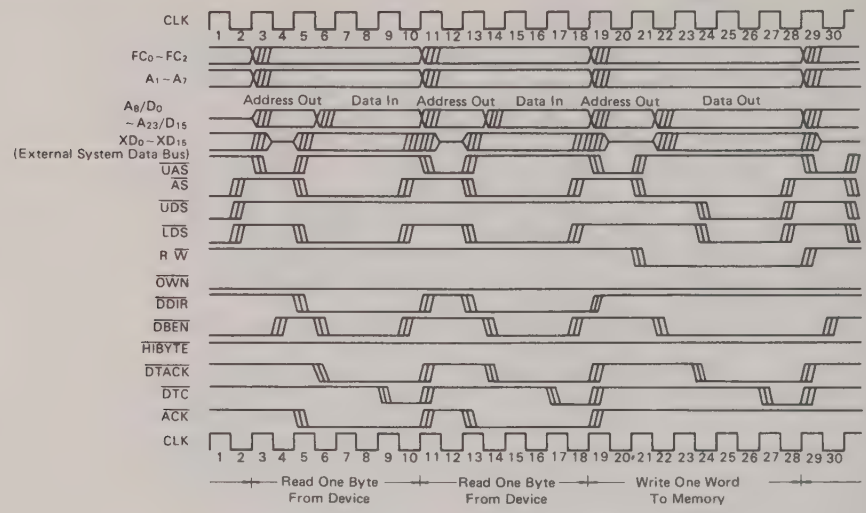

CLK
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
FC0 ~ FC2
A1 ~ A7
A8/D0 ~ A23/D15
XD0 ~ XD15 (External System Data Bus)
Address Out
Data In
Address Out
Data In
Address Out
Data Out
UAS
AS
UDS
LDS
R W
OWN
DDIR
DBEN
HIBYTE
DTACK
DTC
ACK
CLK
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
Read One Byte From Device
Read One Byte From Device
Write One Word To Memory

Figure 23 Dual Addressing Mode, Read/Write Cycle
Source = 8-bit Device, Word Operand

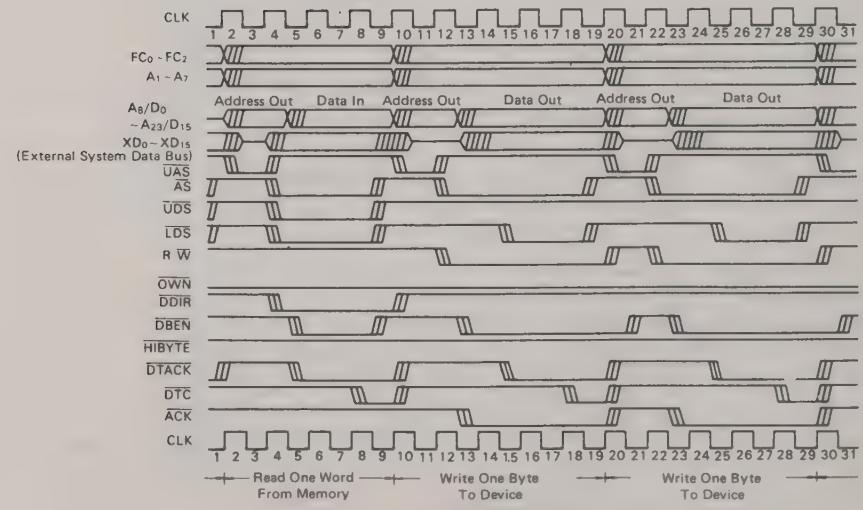

CLK
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
FC0 ~ FC2
A1 ~ A7
A8/D0 ~ A23/D15
XD0 ~ XD15 (External System Data Bus)
Address Out
Data In
Address Out
Data Out
Address Out
Data Out
UAS
AS
UDS
LDS
R W
OWN
DDIR
DBEN
HIBYTE
DTACK
DTC
ACK
CLK
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
Read One Word From Memory
Write One Byte To Device
Write One Byte To Device

Figure 24 Dual Addressing Mode, Read/Write Cycle,
Destination = 8-bit Device, Word Operand

HMCS6800 Compatible Device Transfers

When a channel is programmed to perform HMCS6800 compatible transfers, the $\overline{PCL}$ line for that channel is defined as an Enable clock input. The DMAC performs data transfers between itself and the peripheral device using the HMCS6800 bus protocol, with the $\overline{ACK}$ output providing the $\overline{VMA}$ (valid memory address) signal. Figure 25 illustrates this protocol. Refer to Figure 26 for the read cycle timing and Figure 27 for the write cycle timing. In Figure 26, the DMAC latches the data at the falling edge of clock 19, so a latch to hold the data is necessary as shown in Figure 47.

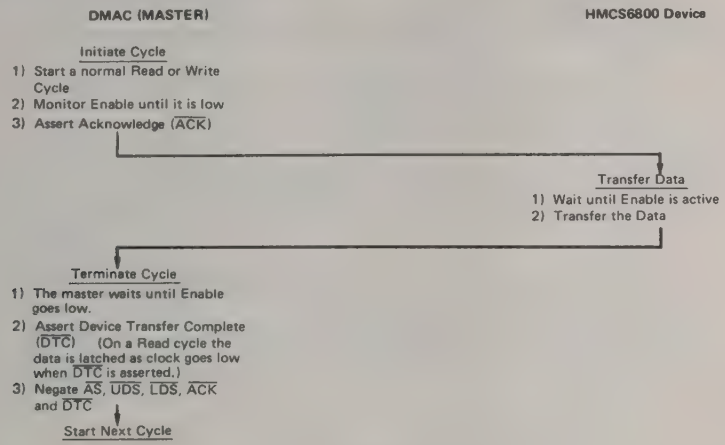

DMAC (MASTER)
HMCS6800 Device
Initiate Cycle
1) Start a normal Read or Write Cycle
2) Monitor Enable until it is low
3) Assert Acknowledge (ACK)
Transfer Data
1) Wait until Enable is active
2) Transfer the Data
Terminate Cycle
1) The master waits until Enable goes low.
2) Assert Device Transfer Complete (DTC) (On a Read cycle the data is latched as clock goes low when DTC is asserted.)
3) Negate AS, UDS, LDS, ACK and DTC
Start Next Cycle

Figure 25 HMCS6800 Cycle Flowchart

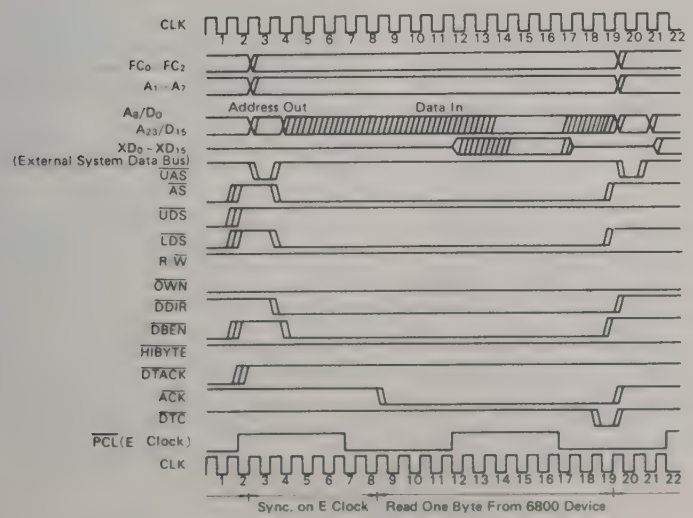


Figure 26 Dual Addressing Mode, HMCS6800 Compatible Device, Read Cycle

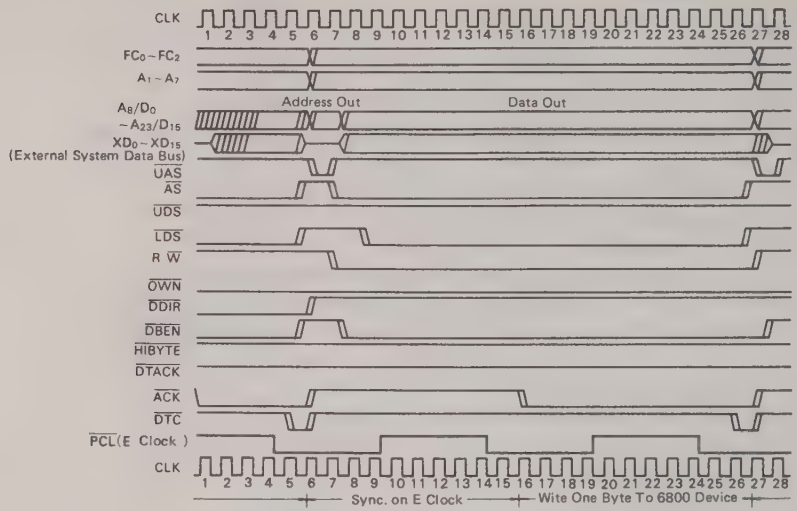

CLK
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28
FC0 ~ FC2
A1 ~ A7
Address Out
Data Out
A8/D0 ~ A23/D15
XD0 ~ XD15 (External System Data Bus)
UAS
AS
UDS
LDS
R W
OWN
DDIR
DBEN
HIBYTE
DTACK
ACK
DTC
PCL (E Clock)
CLK
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28
Sync. on E Clock
Wite One Byte To 6800 Device

Figure 27 Dual Addressing Mode, HMCS6800 Compatible Device, Write Cycle

(2) Single Addressing Mode

Implicitly addressed devices are peripheral devices selected not by address but by $\overline{ACK}$. They do not require addressing of data register during data transfer. Transfers between memory and these devices are controlled by the request/acknowledge protocol. Such peripherals require only one bus cycle to transfer data, and the DMAC internal holding register is not used. Because only the memory is addressed during a data transfer and a transfer done in only on bus cycle, this protocol is called single-address.

Device with $\overline{ACK}$ Transfers

Under this protocol, the communication between peripheral device and the DMAC is performed with a two signal $\overline{REQ}/\overline{ACK}$ handshake. When a request is generated using the request method programmed in the DMAC's internal control registers, the DMAC obtains the bus and responds with $\overline{ACK}$. The DMAC asserts all the bus control signals required for the memory access. Refer to Figure 28 for the flowchart of the data transfer from memory to the device with $\overline{ACK}$. Figure 29 shows the flowchart of the data transfer from the device with $\overline{ACK}$ to memory. When a request is generated using the request method programmed in the control registers, the DMAC obtains the bus and responds with acknowledge. The DMAC asserts all HMCS68000 bus control signals needed for the transfer. When the DMAC accepts $\overline{DTACK}$ from memory, it asserts $\overline{DTC}$ and informs the peripheral device of the transfer termination. Figure 30 and 31 show the transfer timings of the device with $\overline{ACK}$: the port size for the former figure is 8-bit and the latter is 16-bit respectively.

When the transfer is from memory to a device, data is valid when $\overline{DTACK}$ is asserted and remains valid until the data strobes are negated. The assertion of $\overline{DTC}$ from the DMAC may be used to latch the data.

When the transfer is from device to memory, data must be valid on the HMCS68000 bus before the DMAC asserts the data strobes. The data strobes are asserted one clock period after $\overline{ACK}$ is asserted. When the DMAC obtains the bus and starts a DMA cycle, the tri-state of the $\overline{OWN}$ line is cancelled a half clock earlier than other control lines. If the DMA Cycle terminates and the DMAC relinquishes the bus, all the control signals get tri-stated a half clock before $\overline{OWN}$. The $\overline{DDIR}$ and $\overline{DBEN}$ lines are not asserted in the single addressing mode. Four clocks cycle is the smallest bus cycle for the transfer from memory to device. Five clocks cycle is the smallest bus cycle for the transfer from device to memory. If the device port size is 8-bit, either $\overline{LDS}$ or $\overline{UDS}$ is asserted. In the single adressing mode, A_8-A_{23} are outputted for only one and a half clock from the beginning of the DMA bus cycle. Therefore, A_8 through A_{23} needs to be latched externally just like in the dual addressing mode.

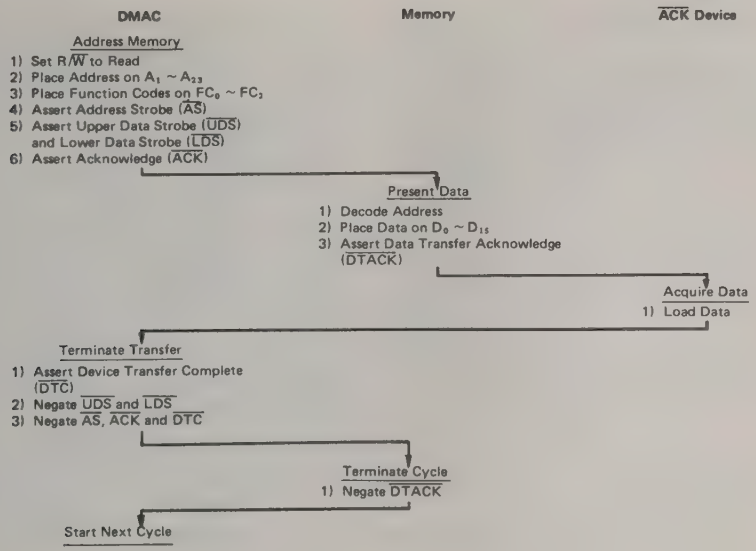

DMAC
Memory
ACK Device
Address Memory
1) Set R/W to Read
2) Place Address on A1 ~ A23
3) Place Function Codes on FC0 ~ FC2
4) Assert Address Strobe (AS)
5) Assert Upper Data Strobe (UDS) and Lower Data Strobe (LDS)
6) Assert Acknowledge (ACK)
Present Data
1) Decode Address
2) Place Data on D0 ~ D15
3) Assert Data Transfer Acknowledge (DTACK)
Acquire Data
1) Load Data
Terminate Transfer
1) Assert Device Transfer Complete (DTC)
2) Negate UDS and LDS
3) Negate AS, ACK and DTC
Terminate Cycle
1) Negate DTACK
Start Next Cycle

Figure 28 Word from Memory to Device with ACK

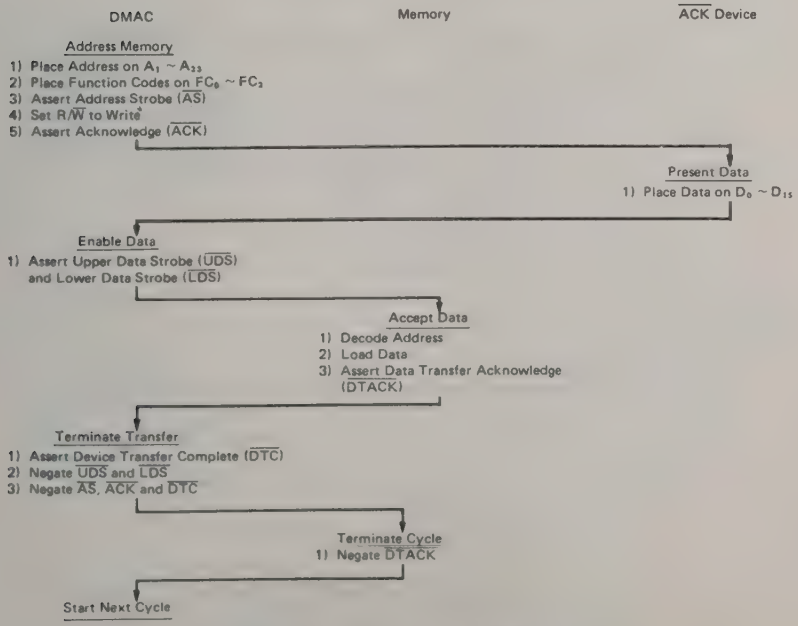

DMAC
Memory
ACK Device
Address Memory
1) Place Address on A1 ~ A23
2) Place Function Codes on FC0 ~ FC2
3) Assert Address Strobe (AS)
4) Set R/W to Write
5) Assert Acknowledge (ACK)
Present Data
1) Place Data on D0 ~ D15
Enable Data
1) Assert Upper Data Strobe (UDS) and Lower Data Strobe (LDS)
Accept Data
1) Decode Address
2) Load Data
3) Assert Data Transfer Acknowledge (DTACK)
Terminate Transfer
1) Assert Device Transfer Complete (DTC)
2) Negate UDS and LDS
3) Negate AS, ACK and DTC
Terminate Cycle
1) Negate DTACK
Start Next Cycle

Figure 29 Word from Device with ACK to Memory

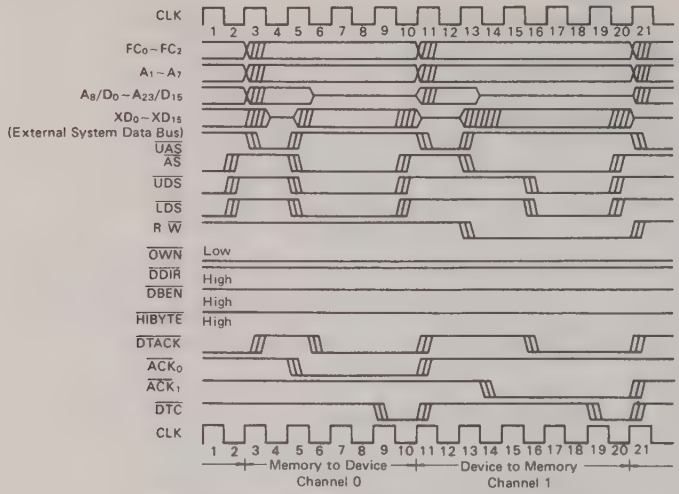


Figure 30 Single Addressing Mode with 16-Bit Devices as Source and Destination (Read-Write Cycles)

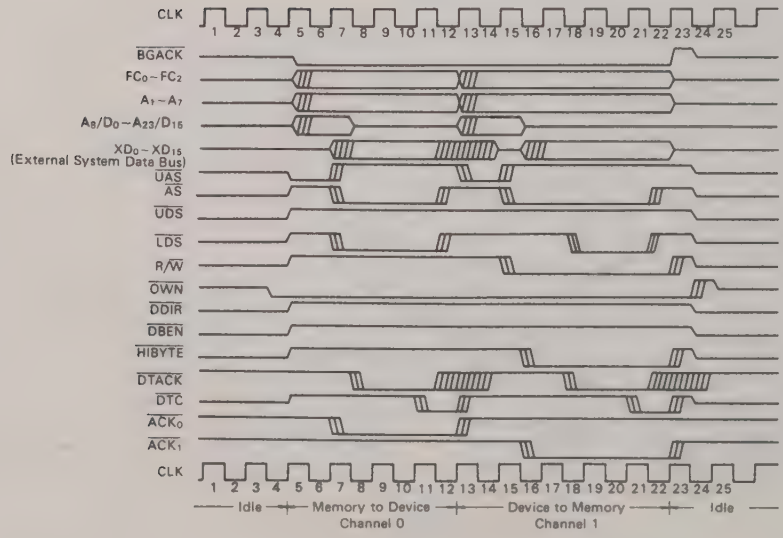


Figure 31 Single Addressing Mode with 8-Bit Device as Source and Destination (Read-Write Cycles)

Device with $\overline{ACK}$ and $\overline{READY}$ Transfers

Under this protocol, the communication between peripheral device and the DMAC is performed using a three signal $\overline{REQ}/\overline{ACK}/\overline{READY}$ handshake. The $\overline{READY}$ input to the DMAC is provided by the $\overline{PCL}$ line. The $\overline{READY}$ line is active low. When a request is generated using the request method programmed in the control registers, the DMAC obtains the bus and asserts $\overline{ACK}$ to notify the device that the transfer is to take place. The DMAC waits for $\overline{READY}$ ($\overline{PCL}$ input), which is a response from the device, in addition to $\overline{DTACK}$ which is a response from memory.

When the DMAC accepts both signals, it terminates the transfer. Refer to Figures 33 and 34 for the flowcharts of the data transfer between memory and the device with $\overline{ACK}$ and $\overline{READY}$. Refer to Figure 35 for the transfer timing of the 8-bit device. When the data transfer is from memory to a device, data is valid from the assertion of $\overline{DTACK}$ to the negation of $\overline{LDS}$ and $\overline{UDS}$. $\overline{DTC}$ is asserted a half clock before $\overline{LDS}$ and $\overline{UDS}$ are negated, so this line may be used for latching the data by the peripheral device. In this case, $\overline{READY}$ ($\overline{PCL}$ input) indicates that the device has received the data. Both $\overline{DTACK}$ and $\overline{READY}$ ($\overline{PCL}$ input) signals are needed for terminating the DMA cycle.

When the data transfer is from the device to memory, data must be valid on the bus before the DMAC asserts $\overline{LDS}$ and $\overline{UDS}$. Therefore, $\overline{READY}$ ($\overline{PCL}$ input) is used as the signal to indicate that the peripheral device has outputted the data on the bus. When the DMAC detects PCL (READY input), then it asserts $\overline{LDS}$ and $\overline{UDS}$. After asserting $\overline{LDS}$ and $\overline{UDS}$, the DMAC terminates the cycle when $\overline{DTACK}$ signal from the memory is detected.

When Array Chain or Link Array Chain is set in Device with $\overline{ACK}$ and $\overline{READY}$ Transfer mode, $\overline{READY}$ input is also necessary during DMA bus cycles for reading the chain information from memory. The circuit as shown in Figure 32 may be used in order to generate $\overline{READY}$ input when reading the chain information from memory.

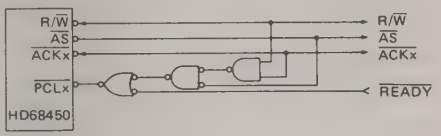


Figure 32 $\overline{READY}$ Circuit When Array or Link Array Chain is set for Device with $\overline{ACK}$ and $\overline{READY}$

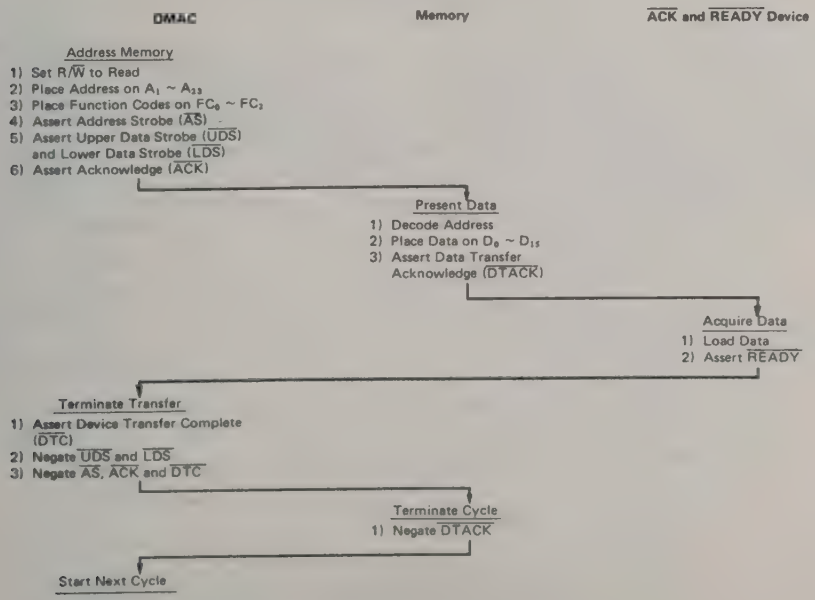

DMAC
Memory
ACK and READY Device
Address Memory
1) Set R/W to Read
2) Place Address on A1 ~ A23
3) Place Function Codes on FC0 ~ FC2
4) Assert Address Strobe (AS)
5) Assert Upper Data Strobe (UDS) and Lower Data Strobe (LDS)
6) Assert Acknowledge (ACK)
Present Data
1) Decode Address
2) Place Data on D0 ~ D15
3) Assert Data Transfer Acknowledge (DTACK)
Acquire Data
1) Load Data
2) Assert READY
Terminate Transfer
1) Assert Device Transfer Complete (DTC)
2) Negate UDS and LDS
3) Negate AS, ACK and DTC
Terminate Cycle
1) Negate DTACK
Start Next Cycle

Figure 33 Word from Memory to Device with $\overline{ACK}$ and $\overline{READY}$

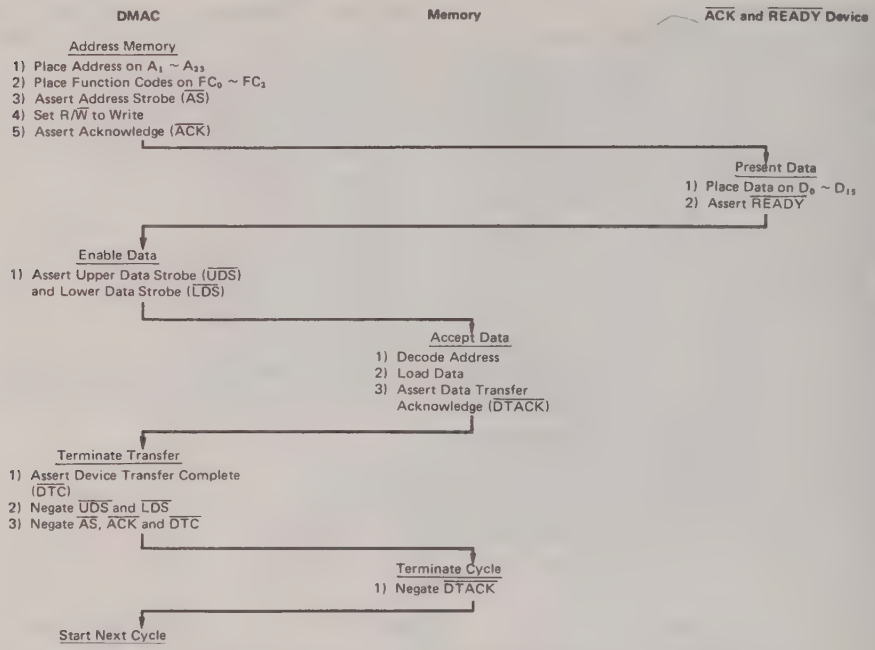

DMAC
Memory
ACK and READY Device
Address Memory
1) Place Address on A1 ~ A23
2) Place Function Codes on FC0 ~ FC2
3) Assert Address Strobe (AS)
4) Set R/W to Write
5) Assert Acknowledge (ACK)
Present Data
1) Place Data on D0 ~ D15
2) Assert READY
Enable Data
1) Assert Upper Data Strobe (UDS) and Lower Data Strobe (LDS)
Accept Data
1) Decode Address
2) Load Data
3) Assert Data Transfer Acknowledge (DTACK)
Terminate Transfer
1) Assert Device Transfer Complete (DTC)
2) Negate UDS and LDS
3) Negate AS, ACK and DTC
Terminate Cycle
1) Negate DTACK
Start Next Cycle

Figure 34 Word from Device with ACK and READY to Memory

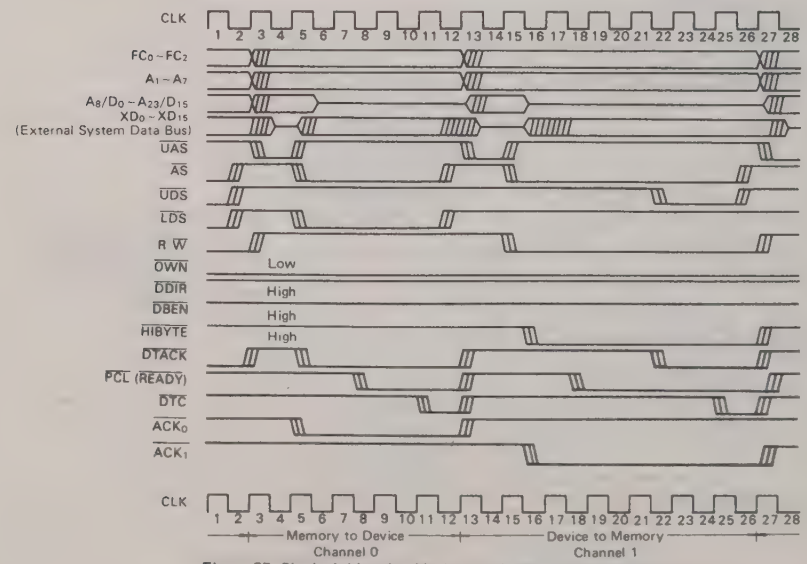


Figure 35 Single Addressing Mode with 8-Bit Devices as Source and Destination with PCL Used as a READY Input (Read-Write Cycles)

Operands and Addressing

Three factors enter into how the actual data is handled: port size, operand size and address sequencing.

Port Size

The DCR is used to program the device port size.

DPS Device Port Size
- 0 8 bit port
- 1 16 bit port

The port size is the number of bits of data which the device can transfer in a single bus cycle. During a DMAC bus cycle, a 16-bit port transfers 16 bits of data on $D_0 \sim D_{15}$, while an 8-bit port transfers 8 bits of data, either on $D_0 \sim D_7$ or on $D_8 \sim D_{15}$. The memory is always assumed to have a port size of 16.

Operand Size

OCR is used to program the operand size.

SIZE Operand Size
- 00 Byte
- 01 Word
- 10 Long word
- 11 (undefined, reserved)

The operand size is the number of bits of data to be transferred to honor a single request. Multiple bus cycles may be required to transfer the operand through the device port. A byte operand consists of 8 bits of data, a word operand consists of 16 bits of data, a long word operand consists of 32 bits of data. The transfer counter counts the number of operands transferred.

Table 2 indicates the combinations supported by the DMAC about the peripheral devices with different port size and operand sizes in the single and dual addressing mode. In the single addressing mode, port size and operand size must be the same. In the dual addressing mode, byte operand cannot be used when the port size is sixteen and the REQG bit is 10 or 11.

Table 2 Operation Combinations

Addressing	Device Type	Port	Operand			REQG bits of OCR
			Byte	Word	Long Word	
Dual	68000, 6800	8	O	O	O	00, 01, 10, 11
Dual	68000, 6800	16	O	O	O	00, 01
Dual	68000, 6800	16	X	O	O	10, 11
Single	with $\overline{ACK}$ or	8	O	X	X	00, 01, 10, 11
	$\overline{ACK}$ & $\overline{READY}$	16	X	O	X	00, 01, 10, 11

O ; enable X ; disable

(3) Address Sequencing

The sequence of addresses generated depends upon the port size, operand size, whether the addresses are to count up, down or not change and whether the transfer is executed in the single addressing mode or the dual addressing mode. The memory address count method and the peripheral device address count method is programmed using the Memory address count (MAC) bit and the Device address count (DAC) bit in the Sequence Control Register (SCR).

(i) Single addressing mode

In the single addressing mode, memory address sequencing is shown in Table 3. If the operand size is byte, the memory address increment is one (1). If the operand size is word, the memory address increment is two (2). If the memory address register does not count, the memory address is unchanged after the transfer.

If the memory address counts up, the increment is added to the memory address; if the memory address counts down, the increment is subtracted from the memory address. The memory address is changed after the operand is transferred.

Table 3 Single Address Sequencing

Port Size	Operand Size	Memory Address Increment		
		+ (increment)	= (unchanged)	- (decrement)
8	Byte	+1	0	-1
16	Word	+2	0	-2

(ii) Dual addressing mode

In the dual addressing mode, the operand size need not match the port size. Thus the transfer of an operand may require several DMA bus cycles. Each DMA bus cycle, between memory and DMAC and between DMAC and the device, is called the operand part and transfers a portion or all of the operand. The addresses of the operand parts are in a linear increasing sequence. The step between the addresses of the operand is two. The size of the operand parts is the minimum of the port size and the operand size. The number of the operand part is the operand size divided by the port size. In the dual addressing mode, memory is regarded as a device whose port size is 16-bits.

If the port size is 16 bits, the operand size is byte, and the request generation method is auto request or auto request at a limited rate, the DMAC packs consecutive transfers. This means that word transfers are made from the associated address with an address increment of two (2). If the initial source address location contains a single byte, the first transfer is a byte transfer to the internal DMAC holding register, and subsequent transfers from the source are word transfers. If the initial destination location contains a single byte, the first transfer is a byte transfer from the internal DMAC holding register, and any remaining byte remains in the holding register. Likewise, if either the final source or destination location contains a single byte, only a byte transfer is done. Packing is not performed if the address does not count; each byte is transferred by a separate access to the same location. The dual address sequencing is shown in Table 4.

Table 4 Dual Address Sequencing

Port Size	Operand Size	Part Size	Operand Part Address	Address Increment		
				+	=	-
8	Byte	Byte	A	+2	0	-2
8	Word	Byte	A, A+2	+4	0	-4
8	Long	Byte	A, A+2, A+4, A+6	+8	0	-8
16	Byte	Pack	A	+P	0	-P
16	Word	Word	A	+2	0	-2
16	Long	Word	A, A+2	+4	0	-4

P = 1 if packing is not done
= 2 if packing is done

Pack = byte if packing is not done
= word if packing is done

An Example of a Dual Address Transfer

This section contains an example of a dual address transfer using Table 4 of Dual-Address Sequencing. The table is reproduced here as Table 5. The transfer mode of this example is the following:

1. Device Port size = 8 bits
2. Operand size = Long Word (32 bits)
3. Memory to Device Transfer
4. Source (Memory) Counts up, Destination (Device) Counts Down
5. Memory Transfer Counter = 2

In this mode, a data transfer from the source (memory) is done according to the 6th row of Table 5, since the port size of the memory is always 16 bits. A data transfer to the destination (device) is done according to the 3rd row of Table 5. Table 6 shows the data transfer sequence.

The memory map of this example is shown in Table 7. The operand consists of BYTE A through BYTE D in memory of Table 7. Prior to the transfer, MAR and DAR are set to 00000012 and 00000108 respectively. The operand is transferred to the 8 bit port device according to the order of transfer number in Table 6.

Table 5 Dual-Address Sequencing (Table 4)

Row No.	Port Size	Operand Size	Operand Part Size	Operand Part Addresses	Address Increment		
					+	=	-
1	8	BYTE	BYTE	A	+2	0	-2
2	8	WORD	BYTE	A, A+2	+4	0	-4
③	8	LONG	BYTE *4	A, A+2, A+4, A+6 *3 *5 *7 *8	+8	0	-8 *10
4	16	BYTE	PACK (BYTE or WORD)**	A	+P	0	-P
5	16	WORD	WORD	A	+2	0	-2
⑥	16	LONG	WORD *2	A, A+2 *1 *6	+4 *9	0	-4

* Numbers in Table 5 correspond to ones in Table 6 and 7.
** Refer to Address Sequencing on Operand Part Size and PACK.

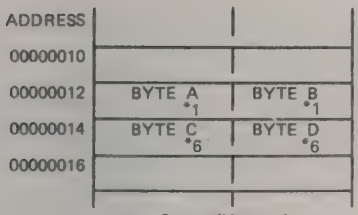
Table 6 An Example of a Data Transfer for One Operand

SRC: Source (Memory), DST Destination (Device), HR: Holding Register (DMAC Internal Reg.)

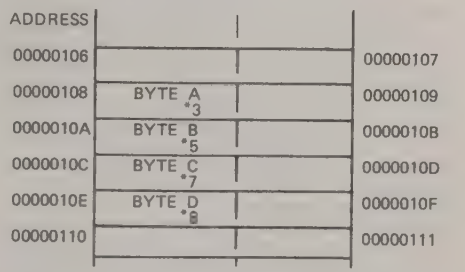
Transfer No.	Data Transfer	Address Output	Data Size on Bus	DMAC Registers after Transfer		Comment
				MAR	DAR	
0	–	–	–	00000012	00000108	Initial Register Setting
1	SRC → HR	00000012 *1	WORD *2	00000014	00000108	Higher order 16 bits of operand is fetched.
2	HR → DST	00000108 *3	BYTE *4	00000014	0000010A	Higher order 16 bits of operand is transferred.
3	HR → DST	0000010A *5	BYTE *4	00000014	0000010C *10	
4	SRC → HR	00000014 *6	WORD *2	00000016 *9	0000010C	Lower order 16 bits of operand is fetched
5	HR → DST	0000010C *7	BYTE *4	00000016	0000010E	Lower order 16 bits of operand is transferred.
6	HR → DST	0000010E *8	BYTE *4	00000016	00000110 *10	
6′	–	–	–	00000016	00000110	MAR, DAR are pointing the next operand addresses when the transfer is complete.

Mode: Port size = 8, Operand size = Long Word, Memory to Device, Source (Memory) Counts Up, Destination (Device) Counts Down

Table 7 Memory Map for the Example of the Data Transfer



Source (Memory)



Destination (Device)

● Initiation and Control of Channel Operation

(1) Operation Initiation

To initiate the operation of a channel the STR bit of the CCR is set to start the operation. Setting the STR bit causes the immediate activation of the channel, the channel will be ready to accept requests immediately. The channel initiates the operation by resetting the STR bit and setting the channel active bit in the CSR. Any pending requests are cleared, and the channel is then ready to receive requests for the new operation. If the channel is configured for an illegal operation, the configuration error is signaled, and no channel operation is run. The illegal operations include the selection of any of the options marked "(undefined, reserved)". If the MTC is set to zero in any operation or BTC is set to zero in the array chaining mode, then the count error is signaled and the channel is not activated. The channel cannot be started if any of the ACT, COC, BTC, NDT or ERR bits is set in the CSR. In this case, the channel signals the operation timing error.

(2) Operation Continuation (Continue Mode)

The continue bit (CNT) allows multiple blocks to be transferred in unchained operations. The CNT bit is set in order to continue the current channel operation. If an attempt is made to continue a chained operation, a configuration error is signaled. The base address register and base transfer counter should have been previously initialized.

The continue bit may be set as the channel is started or while the channel is still active. The operation timing error bit is signaled if a continuation is otherwise attempted.

When the memory transfer counter is exhausted and the continue bit of the CCR is set, the DMAC performs a continuation of the channel operation. The base address, base function code, and base transfer count registers are copied into the memory address, memory function code, and memory transfer count registers. The block transfer complete (BTC) bit of the CSR is set, the continue bit is reset, and the channel begins a new block transfer. If the memory transfer counter is loaded with a terminal count, the count error is signaled.

(3) Operation Halting (Halt)

The CCR has a halt bit which allows suspension of the operation of the channel. If this bit is set, a request may still be generated and recognized, but the DMAC does not attempt to acquire the bus or to make transfers for the halted channel. When this bit is reset, the channel resumes operation and services any request that may have been received while the channel was halted. However, in the burst request mode, the transfer request should be kept asserted until the initiation of the first transfer after clearing the halt bit.

(4) Operation Abort by Software (Software Abort)

Setting the software abort bit (SAB) in the CCR allows the current operation of the channel to be aborted. In this case, the ERR bit and the COC bit in the CSR are set and the ACT bit is reset. The error code for the software abort is set in the CER. The SAB bit is designed to be reset if the ERR bit is reset. When the CCR is read, the SAB always reads as zero(0).

(5) Interrupt Enable

The CCR has an interrupt enable bit (INT) which allows the channel to request interrupts. If INT is set, the channel can request interrupts. If it is clear, the channel will not request interrupts.

- **Channel Operation Termination**

As part of the transfer of an operand, the DMAC decrements the memory transfer counter (MTC). If the chaining mode is not used and the CNT bit is not set or the last block is transferred in the chaining mode, the operation of the channel is complete when the last operand transfer is completed and the MTC is zero. The DMAC notifies the peripheral device of the channel completion via the $\overline{DONE}$ output.

However, in the continue mode, $\overline{DONE}$ is outputted at the termination of every data block transfer. When the channel operation has been completed, the ACT bit of the CSR is cleared, and the COC bit of the CSR is set.

The occurrence of errors, such as the bus error, during the DMA bus cycle also terminates the channel operation. In this case, the ACT bit in the CSR is cleared, the ERR and the COC bits are set, and at the same time the code corresponding to the error that occurred is set in the CER.

(1) Channel Status Register (CSR)

The channel status register contains the status of the channel. The register, except for ACT and PCS bits, is cleared by writing a one (1) into each bit of the register to be cleared. Those bits positions which contain a zero (0) in the write data remain unaffected. ACT and PCS bits are unaffected by the write operation.

COC

The channel operation complete (COC) bit is set if the channel operation has completed. The COC bit must be cleared in order to start another channel operation. The COC bit is cleared only by writing a one to this bit or resetting the DMAC.

PCS

The peripheral status (PCS) bit reflects the level of the $\overline{PCL}$ line regardless of its programmed function. If $\overline{PCL}$ is at "High" level, the PCB bit reads as one. If $\overline{PCL}$ is at "Low" level, the PCS bit reads as zero. The PCS bit is unaffected by writing to the CSR.

PCT

The peripheral control transition (PCT) bit is set, if a falling edge transition has occurred on the $\overline{PCL}$ line. (The $\overline{PCL}$ line must remain at "low" level for at least two clock cycles.) The PCT bit is cleared by writing a one to this bit or resetting the DMAC.

BTC

Block transfer complete (BTC) bit is set when the continue (CNT) bit of CCR is set and the memory transfer counter (MTC) is exhausted. The BTC bit must be cleared before the another continuation is attempted (namely, setting the CNT bit again), otherwise an operation timing error occurs. The BTC bit is cleared by writing a one to this bit or resetting the DMAC.

NDT

Normal device termination (NDT) bit is set when the peripheral device terminates the channel operation by asserting the $\overline{DONE}$ line while the peripheral device was being acknowledged. The NDT bit is cleared by writing a one to this bit or resetting the DMAC.

ERR

Error (ERR) bit is set if any errors have been signaled. When the ERR bit is set, the code corresponding to the kind of the error that occurred is set in the CER. The ERR bit is cleared by writing a one to this bit or resetting the DMAC.

ACT

The active (ACT) bit is asserted after the STR bit has been set and the channel operation has started. This bit is remains set until the channel operation is terminated. The ACT bit is unaffected by write operations. This bit is cleared by the termination of the channel or resetting the DMAC.

(2) Interrupts

The DMAC can signal the termination of the channel operation by generating an interrupt request. The INT bit of the CCR determines if an interrupt can be generated. The interrupt request is generated by the following condition.

① INT = 1

and

② COC = 1 or BTC = 1 or ERR = 1 or NDT = 1 or PCT = 1
(the $\overline{PCL}$ line is an interrupt input)

This may be represented as

$$\overline{IRQ} = INT \cdot (COC + BTC + ERR + NDT + PCT^{*})$$

(*$\overline{PCL}$ line is programmed as an interrupt input.)

When the $\overline{IRQ}$ line is asserted, changing the INT bit from one to zero to one will cause the $\overline{IRQ}$ output to change from "low" to "high" to "low" again. The $\overline{IRQ}$ should be negated by clearing the COC, the BTC, the ERR, the NDT and the PCT bits.

If the DMAC receives $\overline{IACK}$ from the MPU during asserting the $\overline{IRQ}$, the DMAC provides an interrupt vector. If multiple channels have interrupt requests, the determination of which channel presents its interrupt vector is made using the same priority scheme defined for the channel operations.

The bus cycle in which the DMAC provides the interrupt vector when receiving an $\overline{IACK}$ from the MPU is called the interrupt acknowledge cycle. The interrupt vector returned to the MPU comes from either the normal or the error interrupt vector register. The normal interrupt register is used unless the ERR bit of CSR is set, in which case the error interrupt vector register is used. The content of the interrupt vector register is placed on $D_0 \sim D_7$, and $\overline{DTACK}$ is asserted to indicate that the vector is on the data bus. If a reset occurs, all interrupt vector registers are set to $0F (binary 00001111), the value of the uninitialized interrupt vector. The timing of the interrupt acknowledge cycle is shown in Figure 36. The HD68000 MPU outputs the interrupt level into A_1-A_3 and A_4-A_7 is held "high" during the interrupt acknowledge cycle, but the HD68450 DMAC ignores these signals.

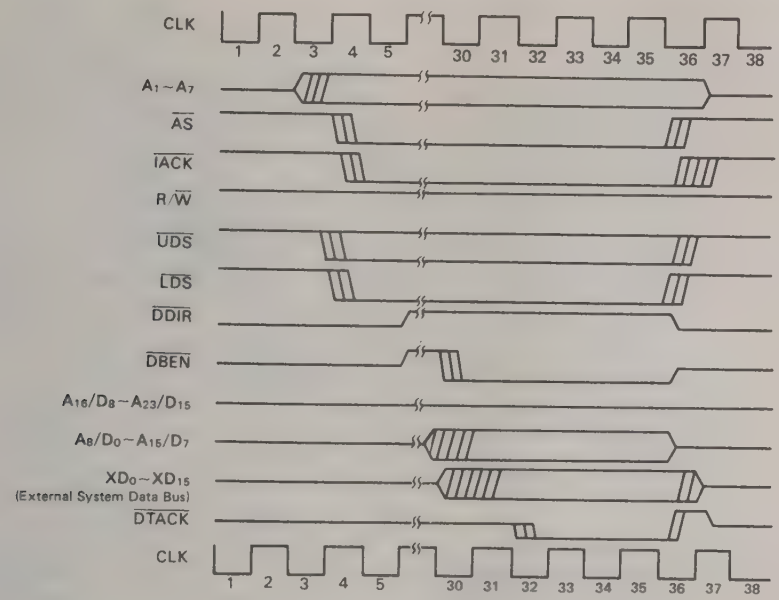


Figure 36 MPU $\overline{IACK}$ Cycle to DMAC

(3) Multiple Data Block Transfer Operation

When the memory transfer counter (MTC) is exhausted, the channel operation still continues if the channel is set to the array chaining mode or the linked array chaining mode and the chain is not exhausted. The channel operation also containes if the continue bit (CNT) of the CCR is set. The DMAC provides the initialization of the memory address register and the memory transfer counter in these cases so that the DMAC can transfer the multiple blocks.

Continued Operation

The continued operation is described in the Initiation and the Control of the Channel Operation section.

Array Chaining

This type of chaining uses an array in memory consisting of memory addresses and transfer counts. Each entry in the array is six bytes long and, consists of four bytes of address followed by two bytes of transfer count. The beginning address of this array is in the base address register, and the number of entries in the array is in the base transfer counter. Before starting any block transfers, the DMAC fetches the entry currently pointed to by the base address register. The address information is placed in the memory address register, and the count information is placed in the memory transfer counter. As each chaining entry is fetched, the base transfer counter is decremented by one. After the chaining entry is fetched, the base address register is incremented to point the next entry. When the base transfer counter reaches a terminal count of zero, the chain is exhausted, and the entry just fetched determines the last block of the channel operation.

An example of the array chaining mode operation and the memory format for supporting for array chaining is shown in Figure 37. The array must start at an even address, or the entry fetch results is an address error. If a terminal count is loaded into the memory transfer counter or the base transfer counter, the count error is signaled. Since the base registers may be read by the MPU, appropriate error recovery information is available should the DMAC encounter an error anywhere in the chain. Contents of the BFC is outputted as the function code when the DMAC is accessing the memory using the base address register. The value of the function code registers are unchanged in the array chaining operation.

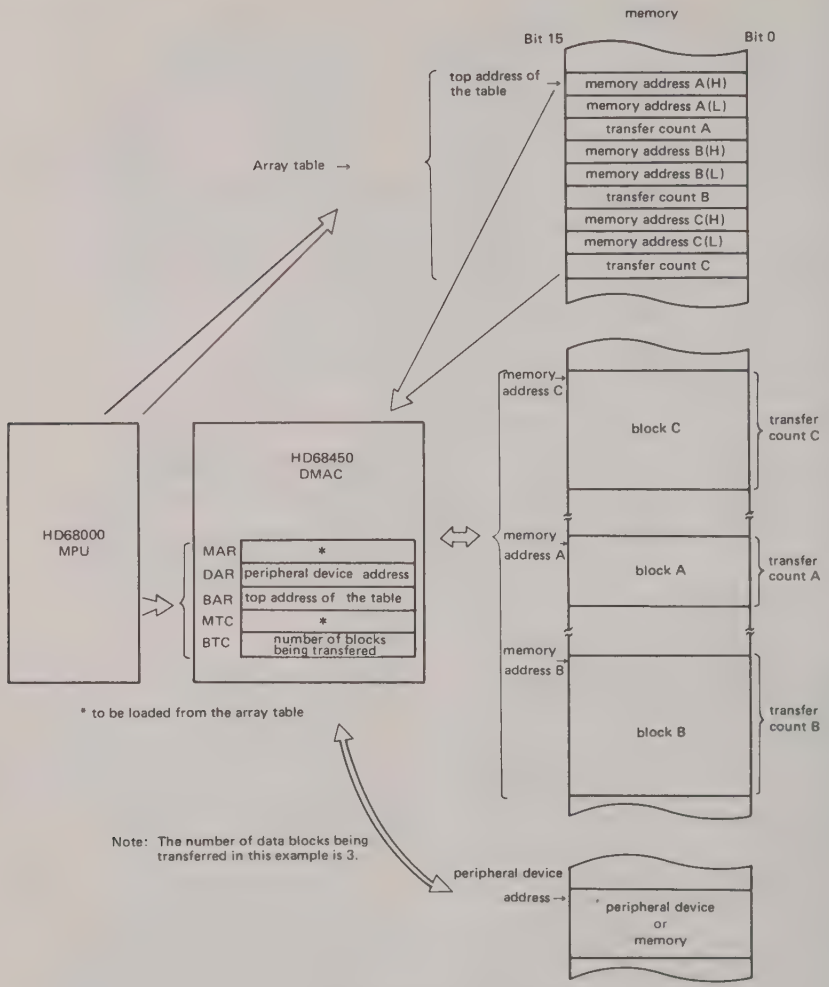


Figure 37 Transfer Example of the Array Chaining Mode

Linked Array Chaining

This type of chaining uses a list in memory consisting of memory address, transfer counts, and link addresses. Each entry in the chain list is ten bytes long, and consists of four bytes of memory address, two bytes of transfer count and four bytes of link address. The address of the first entry in the list is in the base address register, and the base transfer counter is unused. Before starting any block transfers, the DMAC fetches the entry currently pointed to by the base address register. The address information is placed in the memory address register, the count information is placed in the memory transfer counter, and the link address replaces the current contents of the base address register. The channel then begins a new block transfer. As each chaining entry is fetched, the update base address register is examined for the terminal link which has all 32 bits equal to zero. When the new base address is the terminal address, the chain is exhausted, and the entry just fetched determines the last block of the channel operation.

An example of the linked array chaining mode operation and the memory format for supporting it is shown is Figure 38.

In Figure 38, the DMAC transfers data blocks in the order of Block A, Block B, and Block C. In the linked array chaining

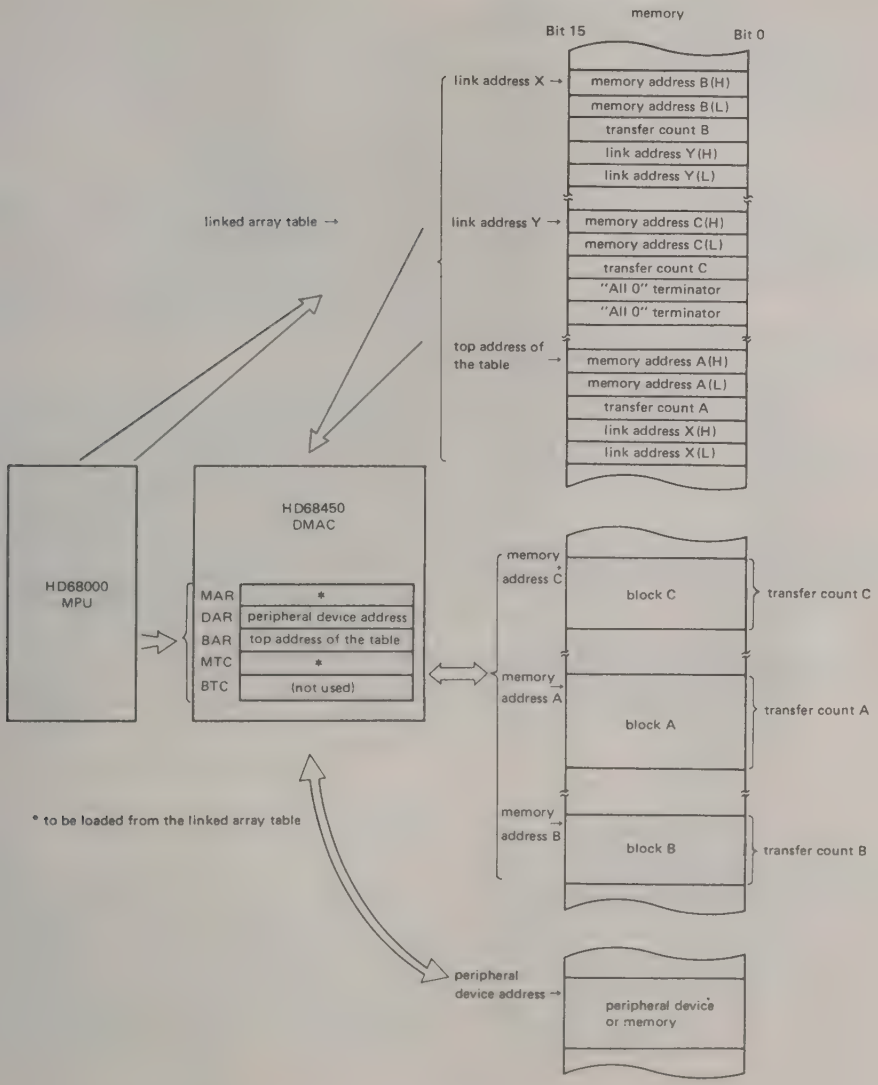


Figure 38 Transfer Example of the Linked Array Chaining Mode

mode, the BTC is not used. When the DMAC refers to the linked array table, the value of the BFC is outputted as the function code. The values of the function code registers are unchanged by the linked array chaining operation.

This type of chaining allows entries to be easily removed or inserted without having to reorganize data within the chain. Since the end of the chain is indicated by a terminal link, the number of entries in the array need not be specified to the DMAC.

The linked array table must start at an even address in the linked array chaining mode. Starting the table at an odd address results in an address error. If "0" is initially loaded to the MTC, the count error is signaled. Because the MPU can read all of the DMAC registers, all necessary error recovery information is available to the operating system.

The comparision of both chaining modes is shown in Table 8.

Table 8 Chaining Mode Address/Count Information

Chaining Mode	Base Address Register	Base Transfer Counter	Completed When
Array Chaining	address of the array table	number of data blocks being transferred	Base Transfer Count = 0
Linked Array Chaining	address of the linked array table	(unused)	Linked Address = 0

(4) Bus Exception Conditions

The DMAC has three lines for inputting bus exception conditions called $\overline{BEC_0}$, $\overline{BEC_1}$, and $\overline{BEC_2}$. The priority encoder can be used to generate these signals externally. These lines are encoded as shown in Table 9.

Table 9

$\overline{BEC_2}$	$\overline{BEC_1}$	$\overline{BEC_0}$	Exception Condition
1	1	1	No exception condition
1	1	0	Halt
1	0	1	Bus error
1	0	0	Retry
0	1	1	Relinquish bus and retry
0	1	0	(undefined, reserved)
0	0	1	(undefined, reserved)
0	0	0	Reset

In order to guarantee, reliable decoding, the DMAC verifies that the incoming code has been statable for two DMAC clock cycles before acting on it. The DMAC picks up $\overline{BEC_0}$-$\overline{BEC_2}$ at the rising edge of the clock. If $\overline{BEC_0}$-$\overline{BEC_2}$ is asserted to the undefined code, the operation of the DMAC does not proceed. For example, when the DMAC is waiting for $\overline{DTACK}$, inputting $\overline{DTACK}$ does not result in the termination of the cycle if $\overline{BEC_0}$-$\overline{BEC_2}$ is asserted to the undefined code. In addition, when the transfer request is received, $\overline{BR}$ is not asserted if the $\overline{BEC_0}$-$\overline{BEC_2}$ is not set to no exception condition.

If exception condition, except for HALT, is inputted during the DMA bus cycle prior to, or in coincidence with $\overline{DTACK}$, the DMAC terminates the current channel operation immediately. Here coincident means meeting the same set up requirements for the same sampling edge of the clock. If a bus exception condition exists, the DMAC does not generate any bus cycles until it is removed. However, the DMAC still recognizes requests.

Halt

The timing diagram of halt is shown in Figure 39. This diagram shows halt being generated during a read cycle from the 68000 compatible device in the dual addressing mode. If the halt exception is asserted during a DMA bus cycle, the DMAC does not terminate the bus cycle immediately. The DMAC waits for the assertion of $\overline{DTACK}$ before terminating the bus cycle so that the bus cycle is completed normally. In the halted state, the DMAC puts all the control signals to high impedance and relinquishes the bus to the MPU. The DMAC does not output the $\overline{BR}$ until halt exception is negated. When halt exception is negated, the DMAC acquires the bus again and proceeds the DMA operation. In order to insure a halt exception operation, the $\overline{BEC}$ lines must be set to halt at least until the assertion of $\overline{DTC}$.

If the DMAC has the bus, but is not executing any bus cycle, the DMAC relinquishes the bus as soon as halt exception is asserted.

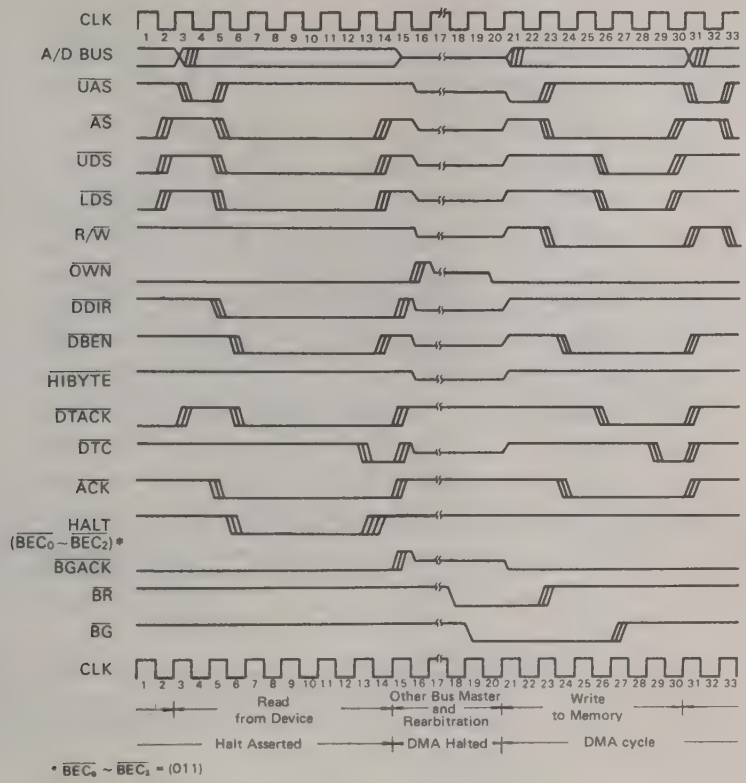
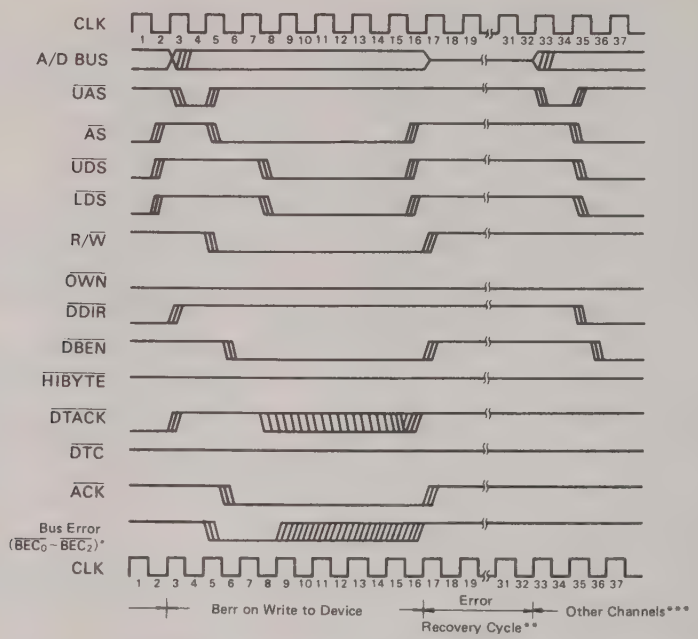


Figure 39 Halt Operation

Bus Error

The bus error exception is generated by external circuitry to indicate the current transfer cannot be successfully completed and is to be aborted. The recognition of this exception during a DMAC bus cycle signals the internal bus error condition for the channel for which the current bus cycle is being run. As soon as the DMAC recognizes the bus error exception, the DMAC immediately terminates the bus cycle and proceeds to the error recovery cycle. In this cycle, the DMAC adjusts the values of the MAR, the DAR, the MTC and the BTC to the values when the bus error exception occurred. 25 clocks are required for the error recovery cycle in the single addressing mode and in the read cycle of the dual addressing mode. 29 clocks are required in the write cycle of the dual addressing mode. If the DMAC does not have any transfer request in the other channels after the error recovery cycle, the DMAC relinquishes the bus.

The diagram of the bus error timing is shown in Figure 40.



* $\overline{BEC_0}$-$\overline{BEC_2}$ = (101)

** In the single addressing mode and in the read cycle of the dual addressing mode: 25 clocks
In the write cycle of the dual addressing mode: 29 clocks

*** The DMAC keeps the bus because the other channels have requests pending. If other channels do not have requests, the DMAC relinquishes the bus after the error recovery cycle.

Figure 40 Bus Error Operation

Retry

The retry exception causes the DMAC to terminate the present operation and retry that operation when retry is removed, and thus will not honor any requests until it is removed. However, the DMAC still recognizes requests. The retry timing is shown in Figure 41.

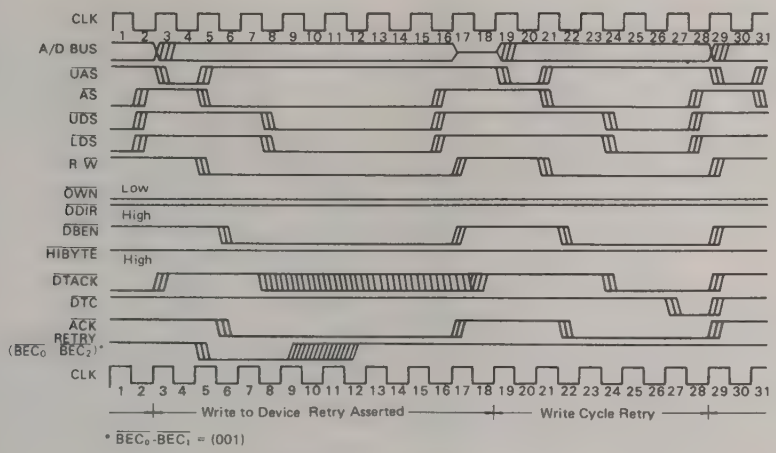


Figure 41 Retry Operation

Relinquish and Retry (R&R)

The relinquish and retry exception causes the DMAC to relinquish the bus and three-state all bus master controls and when the exception is removed, rearbitrate for the bus to retry the previous operation.

The diagram of the relinquish and retry timing is shown in Figure 42.

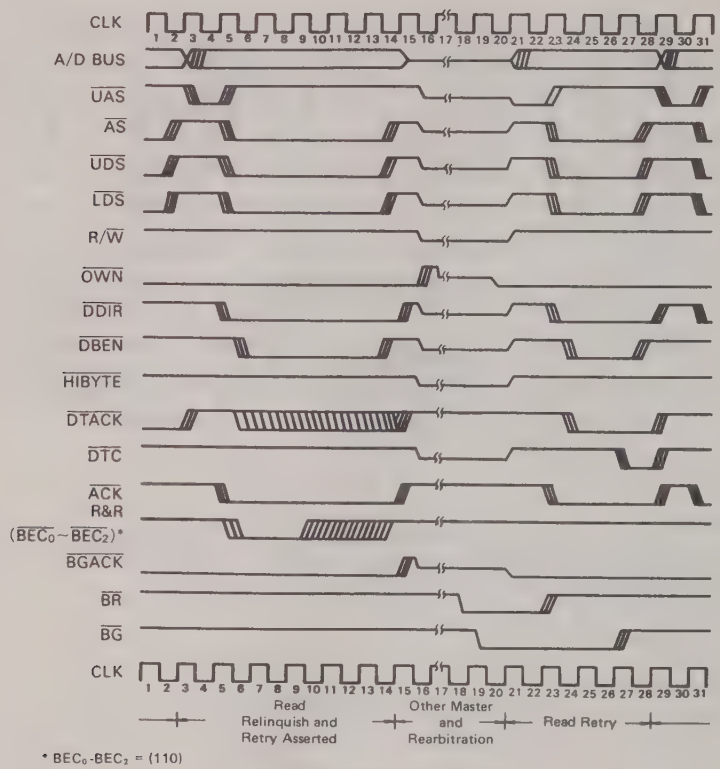


Figure 42 Relinquish and Retry Operation

Reset

The reset provides a means of resetting and initializing the DMAC. If the DMAC is bus master when the reset is asserted, the DMAC relinquishes the bus. Reset clears GCR, DCR, OCR, SCR, CCR, CSR, CPR, and CER for all channels. The NIV and the EIV are all set to $(0F)_{16}$, which is the uninitialized interrupt vector number for the HD68000 MPU. MTC, MAR, DAR, BTC, BAR, MFC, DFC, and BFC are not affected. In order to insure a reset, $\overline{BEC_0} \sim \overline{BEC_2}$ must be kept at "Low" level for at least ten clocks.

(5) Error Conditions

When an error is signaled on a channel, all activity on that channel is stopped. The ACT bit of the CSR is cleared, and the COC bit is set. The ERR bit of the CSR is set, and the error code is indicated in the CER. All pending operations are cleared, so that both the STR and CNT bits of CCR are cleared.

Enumerated below are the error signals and their sources.

(a) Configuration Error – This error occurs if the STR bit is set in the following cases.
  (i) the CNT bit is set at the same time STR bit in the chaining mode.
  (ii) DTYP' specifies a single addressing mode, and the device port size is not the same as the operand size.
  (iii) DTYP specifies a dual addressing mode, DPS is 16 bits, SIZE is 8 bits and REQG is "10" or "11".
  (iv) an undefined configuration is set in the registers. The undefined configurations are: XRM = 01, MAC = 11, DAC = 11, CHAIN = 01, and SIZE = 11.

(b) Operation Timing Error – An operation timing error occurs in the following cases:
  (i) when the CNT bit is set after the ACT bit has been set by the DMAC in the chaining mode, or when the STR and the ACT bits are not set.
  (ii) the STR bit is set when ACT, COC, BTC, NDT or ERR is set.
  (iii) an attempt to write to the DCR, OCR, SCR, MAR, DAR, MTC, MFC, or DFC is made when the STR bit or the ACT bit is set.
  (iv) an attempt to set the CNT bit is made when the BTC and the ACT bits are set.

(c) Address Error – An address error occurs in the following cases:
  (i) an odd address is set for word or long word operands.
  (ii) $\overline{CS}$ or $\overline{IACK}$ is asserted during the DMA bus cycle.

(d) Bus Error – Bus error occurs when a bus error excep-

tion is signaled during a DMA bus cycle.

(e) Count Error – A count error occurs in the following cases:
  (i) The STR bit is set when zero is set in the MTC and the MTC and the chaining mode is not used.
  (ii) the STR bit is set when zero is set in BTC for the array chaining mode.
  (iii) zero is loaded from memory to the BTC or the MTC in the chaining modes or the continue mode.

(f) External Abort – External abort occurs if an abort is asserted by the external circuitry when the $\overline{PCL}$ line is configured as an abort input and the STR or the ACT bit is set.

(g) Software abort – Software abort occurs if the SAB bit is set when the STR or the ACT bit is set.

Error Recovery Procedures

If an error occurs during a DMA transfer, appropriate information is available to the operating system (OS) to allow a software failure recovery operation. The operating system must be able to determine how much data was transferred, where the data was transferred to, an what type of error occurred.

The information available to the operating system consists of the present value of the Memory Address, Device Address and Base Address Registers, the Memory Transfer and Base Transfer Counters, the channel status register, the channel error register, and the channel control register. After the successful completion of any transfer, the memory and device address registers points to the location of the next operand to be transferred and the memory transfer counter contains the number of operands yet to be transferred. If an error occurs during a transfer, that transfer has not completed and the registers contain the values they had before the transfer was attempted. If the channel operation uses chaining, the Base Address Register points to the next chain entry to be serviced, unless the termination occurred while attempting to fetch an entry in the chain. In that case, the Base Address Register points to the entry being fetched. However, in the case of external abort, there are cases in which the previous values are not recovered.

Bus Exception Operating Flow

The bus exception operating flow in the case of multiple exception conditions occurring continuously in sequence is shown in Figure 43. Note that the DMAC can receive and execute the next exception condition. For example, if the retry exception occurs, and next the relinquish and retry exception occurs while the DMAC is waiting for the retry condition to be cleared, the DMAC relinquishes the bus and waits for the exception condition to be cleared. If a bus error occurs during this period, the DMAC executes the bus error exception operation.

The flow diagram of the normal operation without exception operation or errors is shown in Figure 44.

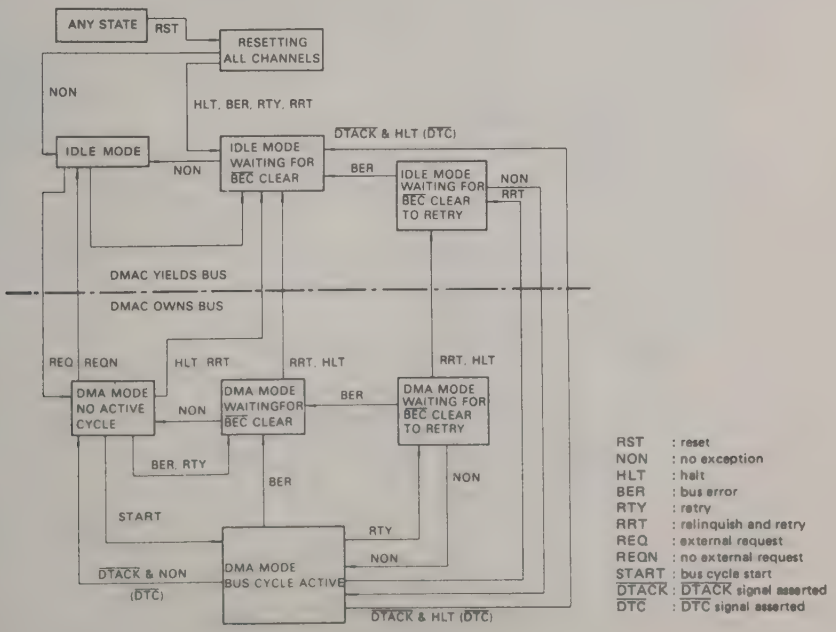


Figure 43 Bus Exception Flow Diagram

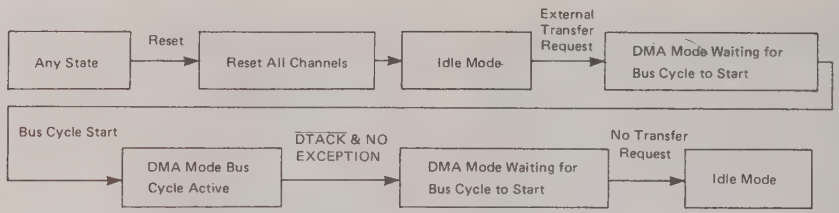


Figure 44 Flow of Normal Operation Without Exception or Error Condition

- **Channel Priorities**

Each channel has a priority level, which is determined by the contents of the Channel Priority Register (CPR). The priority of a channel is a number from 0 to 3, with 0 being the highest priority level. When multiple requests are pending at the DMAC, the channel with the highest priority receives first service. The priority of a channel is independent of the device protocol or the request mechanism for that channel. If there are several requesting channels at the highest priority level, a round-robin resolution is used, that is, as long as these channels continue to have requests, the DMAC does operand transfers in rotation.

Resetting the DMAC puts the priority level of all channels to "0", the highest priority level.

■ APPLICATIONS INFORMATION

Examples of how to interface HD68450 to a HD68000 based system are shown in Figure 45 and Figure 46.

Figure 45 shows an example of how to demultiplex the address/data bus. $\overline{OWN}$ and $\overline{UAS}$ are used to control 74LS373 for latching the address. $\overline{DBEN}$ and $\overline{DDIR}$ are used to control the bi-directional buffer 74LS245.

Figure 46 shows an example of inter-device connection in the HMCS68000 system.

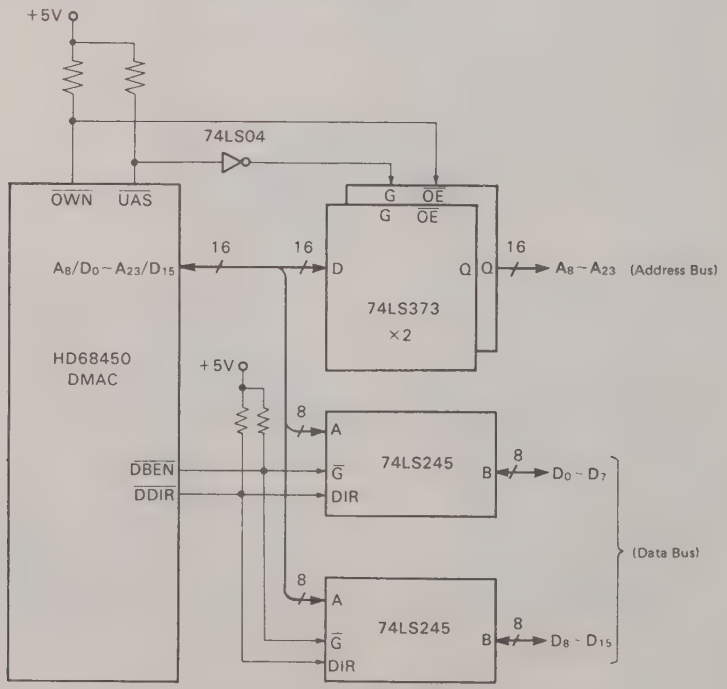
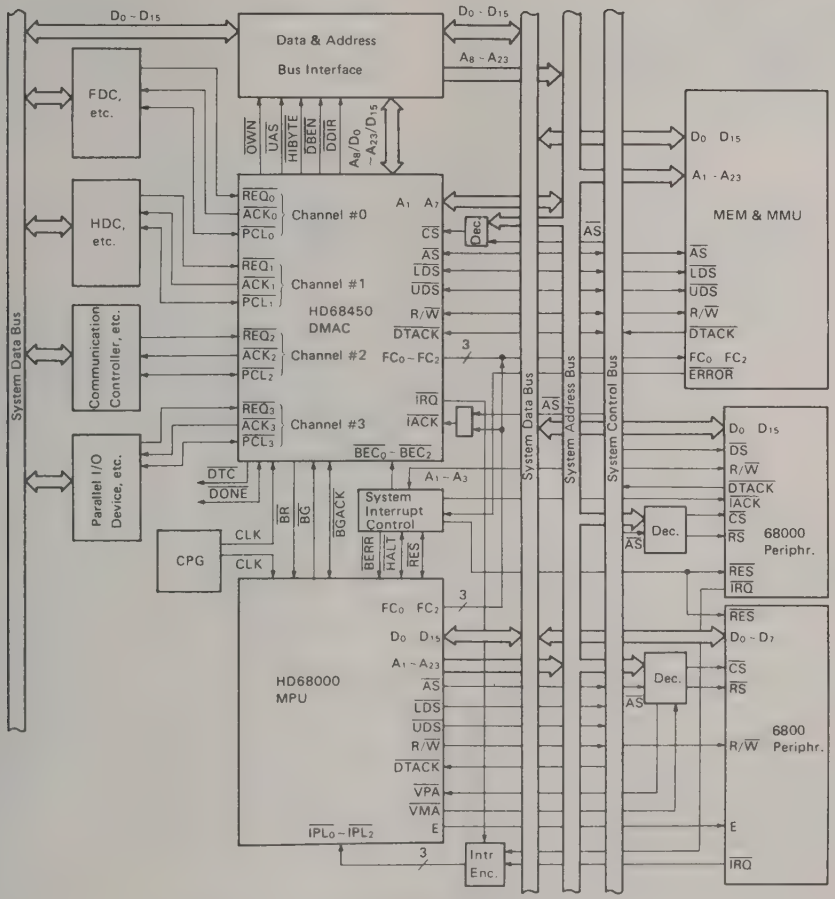


Figure 45 An Example of the Demultiplexed Address Data Bus



The address bus and the system control bus in each device are omitted in this Figure.

Figure 46 An Example of Inter-device Connection in the HMCS68000 System

■ ATTENTION ON USAGE

(1) How to interface various 6800 type peripheral devices to the DMAC based system.

When the DMAC is reading data from the 6800 device, the DMAC latches the data when $\overline{DTC}$ is asserted and not at the falling edge of E clock. The 74LS373 need to be provided externally as shown in Figure 47 so that the data from the 6800 device can be held on the bus for a large period of time until the DMAC can latch the correct data.

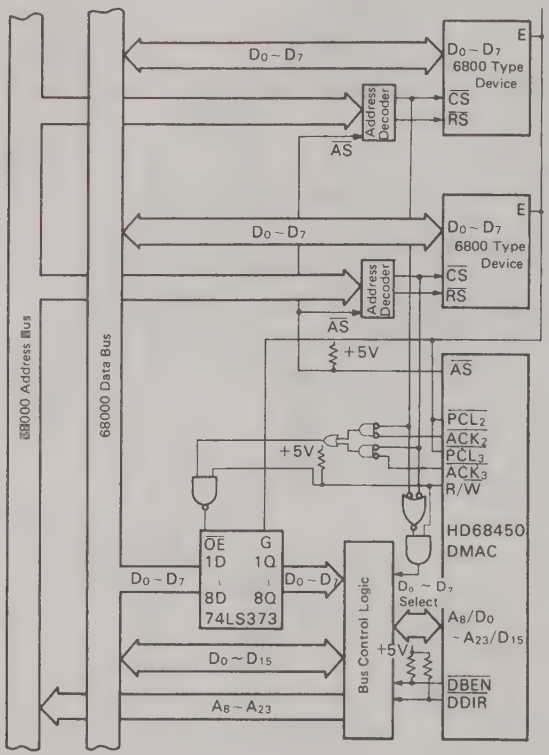


Figure 47 An Example of Connection with 6800 type Peripheral Devices (channel 2 and 3 are used)

(2) When "external abort" is inputted during the $\overline{DONE}$ input cycle

When the transfer direction is from the peripheral device to memory and $\overline{PCL}$ signal is set to the external abort input mode in the dual addressing mode, the external abort will be ignored during the subsequent write cycle from the DMAC's internal holding register to memory if $\overline{DONE}$ is inputted during the read cycle from the peripheral device to the DMAC's internal holding register.

In this case, the channel status register (CSR) and the channel error register (CER) show the normal termination caused by $\overline{DONE}$ Input. The user is able to examine the PCT bit and the ERR bit in order to detect the external abort inputted at the timing described above. If PCT = 1, ERR = 0, and NDT = 1, then an external abort has occurred.

(3) Multiple Errors

The DMAC will log the first error encountered in the channel error resister. If an error is pending in the error register and another error is encountered the second error will not be logged. Even though the second error is not logged in the CER, it will still be recognized internally and the channel will not start.

(4) Relinquish & Retry Exception During Dual Address Mode Operation

When the following two conditions occur simultaneously, incorrect data is outputted by the DMAC at the write cycle immediately following the negation of the relinquish & retry (R&R) exception.

(1) R&R is asserted at the write cycle in the dual address mode.

(2) MPU access to the DMAC's internal register is done after the DMAC relinquishes the bus due to R&R exception.

When the R&R exception occurs during the write cycle of

the dual addressing mode, and the MPU accesses the DMAC's register, then the DMAC's proper operation sequence should be the following (refer to the Fig. 48):

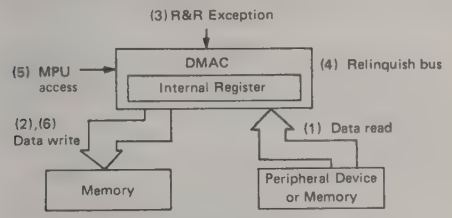


Fig. 48

(1) Data is read by the DMAC during the read cycle
(2) Data read at (1) is outputted at the write cycle
(3) R&R exception is asserted during the write cycle
(4) DMAC relinquishes the bus
(5) MPU accesses the DMAC and it is completed normally
(6) When R&R exception is negated, the DMAC obtains the bus and write cycle is done to output the data read at (1).

But instead, incorrect data is outputted at (6). This is because the data read at (1), which is held internally by the DMAC, is destroyed at (5) when the MPU accesses the DMAC. Avoid occurance of the above condition. For example:

(1) Assert R&R exception only during the read cycle when using dual addressing mode.
(2) If the R&R exception occurred at the write cycle of the dual addressing mode, avoid accessing the DMAC's registers.
(3) Use HALT exception instead of R&R exception to access the DMAC's internal registers.

(5) $\overline{CS}$ Negation Timing

When the $\overline{LDS}$, $\overline{UDS}$ high to $\overline{CS}$ high timing (chip select negation delay) is over 1 clock and the MPU access is long word (32-bit data), then the data stored in the lower word of the register accessed is destroyed and becomes all zeros. In other words, the data in the lower word of the read access is lost and cleared to all zeros. This does not always occur, but on occasional basis due to the asynchronous input timing of the CS signal.

Please observe the timing specification shown in Fig. 49.

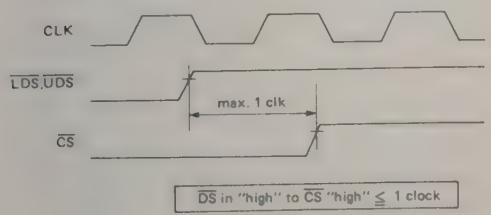


Fig. 49

(6) Unused Function Code (FC_0–FC_2) Lines

When the FC_0, FC_1, and FC_2 lines are not used, please keep these lines "high" by using a pull-up resistor. If these lines are left unconnected, the HD68450 DMAC may not operate properly.

(7) The use of thick wiring is recommended between Vss of the HD68450 and the ground of the circuit board. When a socket is used to install the DMAC on the board, please make sure that the contact of the Vss pins are made well.

• DIFFERENCES AND COMPATIBILITY AMONG THE MASK VERSIONS

• Marking

"OLD" MASK VERSION

S MASK VERSION

NEW U MASK VERSION

• The "OLD" Mask

For the "Old" mask version of the HD68450 DMAC, please refer to the following material, which may be obtained from the sales office:

Microcomputer Device/System Technical Information
No. T-026
"Anomalies of the HD68450 DMAC" Aug. 26, 1983

• The "S" Mask

With the S mask version of the HD68450 DMAC, the anomalies listed in the "Microcomputer Device/System Technical Information, No. T-026 – Anomalies of the HD68450 DMAC" (dated Aug. 26, 1983) is fixed with the exception of item #3 – "Extra data transfer in the burst mode." For the description of the S mask's anomaly, please contact the sales office.

The other remaining anomaly in the "S" mask version – one byte of transfer data is left in the DMAC.

When the DMAC is set to dual addressing mode, port size 8 bits, external request mode, and data transfer from peripheral device to memory, the last byte of the transfer may be left inside the DMAC's internal register without being transferred to memory if the transfer is stopped before the transfer count is exhausted. The last byte that is left inside the DMAC becomes unaccessable by the MPU.

In this mode, the DMAC transfers data repeating the following bus cycles:

(1) READ BYTE
(Byte is read from the peripheral device to DMAC)

(2) READ BYTE
(Byte is read from the peripheral device to DMAC)

(3) WRITE WORD
(Word is written to memory from DMAC)

If the transfer is terminated after (1) READ BYTE (see NOTE*), then the byte data that was ready by (1) READ BYTE bus cycle is not written to memory and is left inside the DMAC's internal holding register. The DMAC's internal holding register cannot be accessed by the MPU, so that it becomes "lost".

This will not occur when single addressing mode is used. So, please use the single addressing mode when the transfer needs to be terminated before the transfer is exhausted.

NOTE*: The methods to terminate the transfer operation before the transfer counter becomes zero are (1) assert external abort using the $\overline{PCL}$. (2) set the SAB bit to cause software abort.

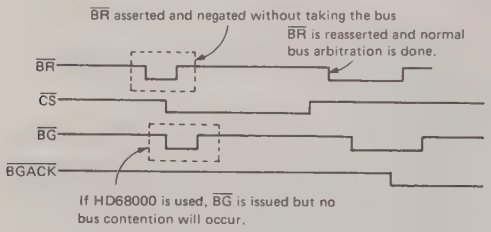
- **The New "U" Mask**

The remaining anomaly of the S mask -- "Extra data transfer in the burst mode" is fixed in the U mask. In order to correct the remaining anomaly of the S mask-one byte of transfer data is left in the DMAC, the U mask has a new mode operation that repeats READ BYTE – WRITE BYTE operation in the dual addressing mode, port size 8 bits, external request mode, and data transfer from peripheral device to memory. The new mode uses the SIZE = 11 in the operation control register. In the "OLD" mask and the S mask version, SIZE = 11 causes an configuration error.

- **Bus Arbitration Problem in the S and U mask version**

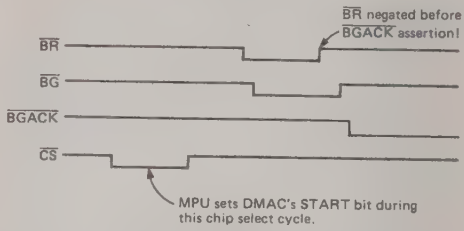
$\overline{BR}$ Negation on MPU's $\overline{CS}$

If the MPU asserts DMAC's $\overline{CS}$ when the DMAC has its $\overline{BR}$ asserted, then the DMAC negates its $\overline{BR}$. This is shown as follows:


BR asserted and negated without taking the bus
BR is reasserted and normal bus arbitration is done.
BR
CS
BG
BGACK
If HD68000 is used, BG is issued but no bus contention will occur.

$\overline{BR}$ negation before $\overline{BGACK}$ assertion

When the MPU sets the START bit of the DMAC, and a different channel is already active in the limited-rate auto-request mode, then the following bus arbitration timing may occur.



In this timing, the $\overline{BR}$ is negated too early, e.g. before the $\overline{BGACK}$ is asserted. This will cause bus contention between the DMAC and the MPU. For the description of these problems, please contact the sales office.

HD63463-4, HD63463-6, HD63463-8
HDC (Hard Disk Controller)

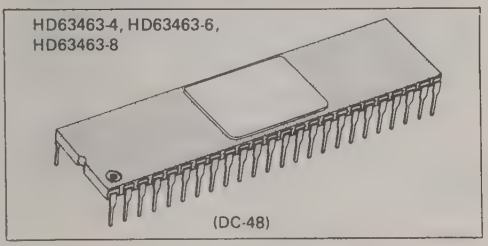
–ADVANCE INFORMATION–

The Hard Disk Controller (HDC) is a CMOS VLSI microcomputer peripheral device capable of controlling Winchester type hard disk drives. The HDC is also a new generation hard disk controller that controls several types of disk drive classified as follows.

(a) Interface type ; ST506 interface
SMD interface (Storage Module Device interface)

(b) Media Format ; Hard sector
Soft sector

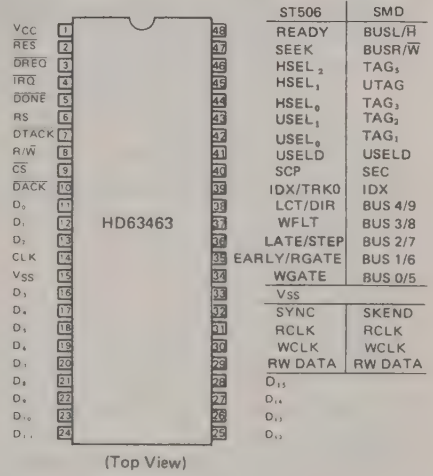
The control functions are greatly enhanced by the presence of powerful high level command and internal two Data Buffers. Therefore, the HDC can be applied to many applications with less CPU software overhead and it enhances system throughput.



■ FEATURES

- Internal data buffer ; 512 bytes (256 bytes x 2)
- High level commands; RECALIBRATE, SEEK, COMPARE DATA, READ DATA, WRITE DATA, FIND ID, READ ID, WRITE FORMAT, CHECK DATA, MEMORY TO BUFFER, BUFFER TO MEMORY etc.
- Interfaceable drives
  - HDD interface : ST506, SMD
  - Sector : Soft-sector, hard-sector
  - Serial data code : MFM, NRZ
- Track format
  - Tracks per side : Max. 1024 tracks
  - Sectors per track : Max. 256 sectors
  - Bytes per sector : 256, 512, 1024, 2048, 4096 bytes
- Error check & correction
  - 16 bit CRC
  - 32 bit fire code : Automatic error correction
- Multiple sector/Multiple track data transfer capability
- Parallel seek/Normal seek operation
- Control up to ; 4 drives with 8 heads (ST506)
  8 drives with 32 heads (SMD)
- DMA interface
- Diagnostic capability: Set high impedance state
- CMOS +5 V single power supply

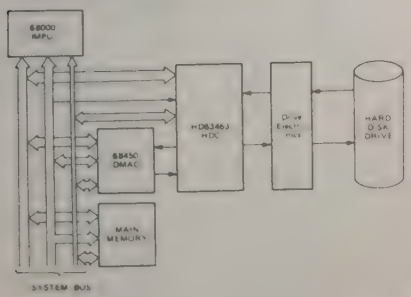
■ PIN ARRANGEMENT



ST506	SMD
READY	BUSL/H
SEEK	BUSR/W
HSEL 2	TAG3
HSEL 1	UTAG
HSEL 0	TAG3
USEL 1	TAG2
USEL 0	TAG1
USELD	USELD
SCP	SEC
IDX/TRK0	IDX
LCT/DIR	BUS 4/9
WFLT	BUS 3/8
LATE/STEP	BUS 2/7
EARLY/RGATE	BUS 1/6
WGATE	BUS 0/5
V_{SS}	
SYNC	SKEND
RCLK	RCLK
WCLK	WCLK
RW DATA	RW DATA

(Top View)

■ SYSTEM CONFIGURATION



■ TYPE OF PRODUCTS

HDC	CLK Frequency	RCLK/WCLK Frequency
HD63463-4	4 MHz	10 MHz
HD63463-6	6 MHz	15 MHz
HD63463-8	8 MHz	20 MHz

HD63484-4, HD63484-6, HD63484-8

ACRTC (Advanced CRT Controller)

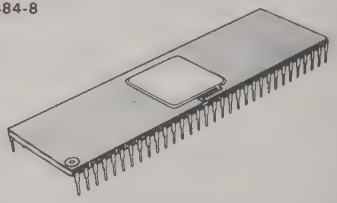
—PRELIMINARY—

The Advanced CRT Controller (ACRTC) is a CMOS VLSI microcomputer peripheral device capable of controlling raster scan type CRTs to display both graphics and characters. The ACRTC is also a new generation CRT controller that is based on a bit-mapped technology and has more display control functions than those of an HD6845S (CRTC).

The ACRTC prepares the mechanisms to use at one of three modes; character only, graphic only and multiplexed character/graphic modes. Therefore, the ACRTC can be applied to many applications, from character only display devices to large full-graphic systems, as the key devices.

The ACRTC can reduce a CPU software overhead and enhance system throughput.

HD63484-4, HD63484-6, HD63484-8



(DC-64)

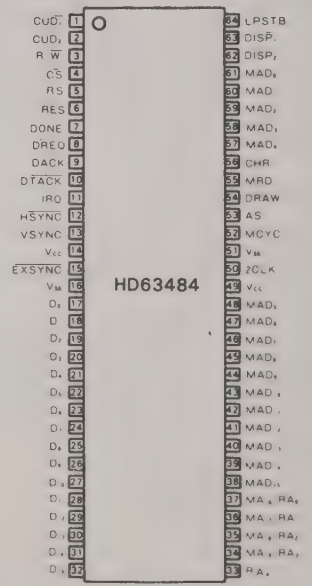
■ FEATURES

- High speed graphic drawings
  - · Drawing rate : Maximum 500 ns/pixel (Color drawing)
  - · Drawn graphics : Dot, Line, Rectangle, Poly-line, Poly-gon, Circle, Ellipse, Paint, Copy, etc.
  - · Drawn colors : 16-bit/word
    1-, 2-, 4-, 8-, 16-bit/pixel (5 types)
    monochrome to max. 64k colors.
- Large frame memory space
  - · Maximum 2M bytes graphic memory
    128k bytes character memory
    separated from the MPU memory
  - · Available to maximum 4096 x 4096 high-resolution CRT (1 bit/pixel mode)
- Various CRT display controls
  - · Split screens (3 displays and 1 window)
  - · Zooming up (1 to 16 times)
  - · Scroll (Vertical and horizontal)
- External synchronization
  - · Synchronization between ACRTCs or between the ACRTC and external device (ex. TV system or other controller)
- DMA interface
- Two programmable cursors
- Three scan modes
  - · Non-interlace, Interlace Sync. and Interlace Sync. & Video modes
- Interrupt request to MPU
- 256 characters/line, 32 rasters/line, 4096 rasters/screen
- Maximum clock frequency 8 MHz
- CMOS, +5V single power supply

■ TYPE OF PRODUCTS

ACRTC	Clock Frequency (2CLK)
HD63484-4	4 MHz
HD63484-6	6 MHz
HD63484-8	8 MHz

■ PIN ARRANGEMENT



(Top View)

■ ABSOLUTE MAXIMUM RATINGS

Item	Symbol	Rating	Unit
Supply Voltage	V_{CC}*	−0.3~+7.0	V
Input Voltage	V_{in}*	−0.3~V_{CC}+0.3	V
Allowable Output Current	$\|I_O\|$**	5	mA
Total Allowable Output Current	$\|\Sigma I_O\|$***	120	mA
Operating Temperature	T_{opr}	0~+70	°C
Storage Temperature	T_{stg}	−55~+150	°C

* This value is in reference to Vss = 0V.
** The allowable output current is the maximum current that may be drawn from, or flow out to, one output terminal or one input/output common terminal.
*** The total allowable output current is the total sum of currents that may be drawn from, or flow out to, output terminals or input/output common terminals.
(Note) Using an LSI beyond its maximum ratings may result in its permanent destruction. LSI's should usually be used under recommended operating conditions. Exceeding any of these conditions may adversely effect its reliability.

■ RECOMMENDED OPERATING CONDITIONS

Item	Symbol	min	typ	max	Unit
Supply Voltage	V_{CC}*	4.75	5.0	5.25	V
Input "Low" Level Voltage	V_{IL}*	0	–	0.7	V
Input "High" Level Voltage	V_{IH}*	2.2	–	V_{CC}	V
Operating Temperature	T_{opr}	0	25	70	°C

* This value is in reference to Vss = 0V.

■ **ELECTRICAL CHARACTERISTICS**

• **DC CHARACTERISTICS (V_{CC} = 5.0V ± 5%, V_{SS} = 0V, Ta = 0 to +70°C unless otherwise noted)**

Item		Symbol	Measuring Condition	4 MHz Version HD63484-4		6 MHz Version HD63484-6		8 MHz Version HD63484-8		Unit
				min	max	min	max	min	max	
Input "High" Level Voltage	All Inputs	V_{IH}		2.2	V_{CC}	2.2	V_{CC}	2.2	V_{CC}	V
Input "Low" Level Voltage	All Inputs	V_{IL}		−0.3	0.7	−0.3	0.7	−0.3	0.7	V
Input Leak Current	R/$\overline{W}$, $\overline{CS}$, RS, $\overline{RES}$, $\overline{DACK}$, 2CLK, LPSTB	I_{in}	V_{in}=0~V_{CC}	−2.5	2.5	−2.5	2.5	−2.5	2.5	μA
Three State (Off State) Input Current	D_0~D_{15}, $\overline{EXSYNC}$, MAD_0~MAD_{15}	I_{TSI}	V_{in}=0.4~V_{CC}	−10	10	−10	10	−10	10	μA
Output "High" Level Voltage	D_0~D_{15}, MAD_0~MAD_{15}, $\overline{CUD_1}$, $\overline{CUD_2}$, $\overline{DREQ}$, $\overline{DTACK}$, $\overline{HSYNC}$, $\overline{VSYNC}$, $\overline{EXSYNC}$,	V_{OH}	I_{OH}=−400μA	V_{CC} −1.0	–	V_{CC} −1.0	–	V_{CC} −1.0	–	V
Output "Low" Level Voltage	$\overline{DISP_1}$, $\overline{DISP_2}$, CHR, MRD, $\overline{DRAW}$, $\overline{AS}$, MCYC, RA_4, MA_{18}/RA_0~MA_{19}/RA_3	V_{OL}	I_{OL}=2.2mA	–	0.5	–	0.5	–	0.5	V
Output Leak Current (Off State)	$\overline{IRQ}$, $\overline{DONE}$	I_{LOH}	V_{OH}=V_{CC}	–	10	–	10	–	10	μA
Input Capacity	D_0~D_{15}, $\overline{EXSYNC}$, MAD_0~MAD_{15}, R/$\overline{W}$, $\overline{CS}$, RS, $\overline{RES}$, $\overline{DACK}$, 2CLK, LPSTB	C_{in}	V_{in}0V, Ta=25°C, f=1.0MHz	–	15	–	15	–	15	pF
Output Capacity	$\overline{IRQ}$, $\overline{DONE}$	C_{out}	V_{in}=0V, Ta=25°C, f=1.0MHz	–	15	–	15	–	15	pF
Current Consumption		I_{CC}	· Chip not selected · Display in progress	–	60	–	80	–	100	mA
			· Data bus in read/write operation · Display in progress · Command execution in progress	–	60	–	80	–	100	

● **AC CHARACTERISTICS (V_{CC} = 5.0V ± 5%, V_{SS} = 0V, Ta = 0 to +70°C unless otherwise noted)**

No.	Item	Symbol	Measuring Condition	4 MHz Version HD63484-4		6 MHz Version HD63484-6		8 MHz Version HD63484-8		Unit
				min	max	min	max	min	max	
	Frequency of Operation	f		1	4	1	6	1	8	MHz
1	Clock Cycle Time	t_{cyc}	Fig. 1	250	1000	167	1000	125	1000	ns
2	Clock "High" Level Pulse Width	t_{PWCH}		115	500	75	500	55	500	ns
3	Clock "Low" Level Pulse Width	t_{PWCL}		115	500	75	500	55	500	ns
4	Clock Rise Time	t_{Cr}		–	10	–	10	–	10	ns
5	Clock Fall Time	t_{Cf}		–	10	–	10	–	10	ns
6	R/$\overline{W}$ Setup Time	t_{RWS}	Fig. 2, Fig. 3	70	–	60	–	50	–	ns
7	R/$\overline{W}$ Hold Time	t_{RWH}		0	–	0	–	0	–	ns
8	RS Setup Time	t_{RSS}		70	–	60	–	50	–	ns
9	RS Hold Time	t_{RSH}		0	–	0	–	0	–	ns
10	$\overline{CS}$ Setup Time	t_{CSS}		50	–	40	–	40	–	ns
11	$\overline{CS}$ "High" Level Width	t_{WCSH}		80	–	70	–	60	–	ns
12										
13	Read Wait Time	t_{RWAI}	Fig. 2	0	–	0	–	0	–	ns
14	Read Data Access Time	t_{RDAC}		–	120	–	100	–	80	ns
15	Read Data Hold Time	t_{RDH}		10	–	10	–	10	–	ns
16	Read Data Turn Off Time	t_{RDZ}		–	60	–	60	–	60	ns
17	$\overline{DTACK}$ Delay Time (Z to L)	t_{DTKZL}	Fig. 2, Fig. 3	–	90	–	80	–	70	ns
18	$\overline{DTACK}$ Hold Time (D to L)	t_{DTKDL}	Fig. 2	0	–	0	–	0	–	ns
19	$\overline{DTACK}$ Release Time (L to H)	t_{DTKLH}	Fig. 2, Fig 3	–	100	–	90	–	80	ns
20	$\overline{DTACK}$ Turn Off Time (H to Z)	t_{DTKZ}		–	100	–	100	–	100	ns
21	Data Bus 3 State Recovery Time 1	t_{DBRT1}	Fig. 2	0	–	0	–	0	–	ns
22	Write Wait Time	t_{WWAI}	Fig 3	0	–	0	–	0	–	ns
23	Write Data Setup Time	t_{WDS}		80	–	60	–	40	–	ns
24	Write Data Hold Time	t_{WDH}		10	–	10	–	10	–	ns
25	$\overline{DREQ}$ Delay Time1	t_{DRQD1}	Fig. 4, Fig. 5	–	150	–	130	–	110	ns
26	$\overline{DREQ}$ Delay Time2	t_{DRQD2}		–	90	–	80	–	70	ns
27	DMA R/$\overline{W}$ Setup Time	t_{DRWS}		70	–	60	–	50	–	ns
28	DMA R/$\overline{W}$ Hold Time	t_{DRWH}		0	–	0	–	0	–	ns
29	$\overline{DACK}$ Setup Time	t_{DAKS}		50	–	40	–	40	–	ns
30	$\overline{DACK}$ "High" Level Width	t_{WDAKH}		80	–	70	–	60	–	ns
31										
32	DMA Read Wait Time	t_{DRW}	Fig. 4	0	–	0	–	0	–	ns
33	DMA Read Data Access Time	t_{DRDAC}		–	120	–	100	–	80	ns
34	DMA Read Data Hold Time	t_{DRDH}		10	–	10	–	10	–	ns
35	DMA Read Data Turn Off Time	t_{DRDZ}		–	60	–	60	–	60	ns
36	DMA $\overline{DTACK}$ Delay Time (Z to L)	t_{DDTZL}	Fig. 4, Fig. 5	–	90	–	80	–	70	ns
37	DMA $\overline{DTACK}$ Delay Time (D to L)	t_{DDTDL}	Fig 4	0	–	0	–	0	–	ns
38	DMA $\overline{DTACK}$ Release Time (L to H)	t_{DDTLH}	Fig 4, Fig 5	–	100	–	90	–	80	ns
39	DMA $\overline{DTACK}$ Turn Off Time (H to Z)	t_{DDTHZ}		–	100	–	100	–	100	ns
40	$\overline{DONE}$ Output Delay Time	t_{DND}		–	90	–	80	–	70	ns
41	$\overline{DONE}$ Output Turn Off Time (L to Z)	t_{DNLZ}		–	100	–	90	–	80	ns

(to be continued)

No.	Item	Symbol	Measuring Condition	4 MHz Version HD63484-4		6 MHz Version HD63484-6		8 MHz Version HD63484-8		Unit
				min	max	min	max	min	max	
42	Data Bus 3 State Recovery Time 2	t_{DBRT2}	Fig. 4	0	–	0	–	0	–	ns
43	$\overline{DONE}$ Input Pulse Width	t_{DNPW}	Fig. 4, Fig. 5	2	–	2	–	2	–	Clk. Cyc.
44	DMA Write Wait Time	t_{DWW}	Fig. 5	0	–	0	–	0	–	ns
45	DMA Write Data Setup Time	t_{DWDS}		80	–	60	–	40	–	ns
46	DMA Write Data Hold Time	t_{DWDH}		10	–	10	–	10	–	ns
47										
48	$\overline{AS}$ "Low" Level Pulse Width	t_{PWASL}	Fig. 6~ Fig. 9	80	–	40	–	25	–	ns
49	Memory Address Hold Time 2	t_{MAH2}	Fig. 7, Fig. 8	10	–	10	–	10	–	ns
50	$\overline{AS}$ Delay Time 1	t_{ASD1}	Fig. 6~ Fig. 9	–	90	–	75	–	65	ns
51	$\overline{AS}$ Delay Time 2	t_{ASD2}		–	90	–	75	–	65	ns
52	Memory Address Delay Time	t_{MAD}		–	95	–	80	–	70	ns
53	Memory Address Hold Time 1	t_{MAH1}		10	–	10	–	10	–	ns
54	Memory Address Turn Off Time (A to Z)	t_{MAAZ}	Fig 6, 7 Fig. 9	–	50	–	50	–	50	ns
55	Memory Address Data Setup Time	t_{MRDS}	Fig. 7	60	–	50	–	40	–	ns
56	Memory Read Data Hold Time	t_{MRDH}		10	–	10	–	10	–	ns
57	MA/RA Delay Time	t_{MARAD}	Fig. 6~ Fig. 9	–	100	–	90	–	80	ns
58	MA/RA Hold Time	t_{MARAH}	Fig. 6~ Fig. 8	10	–	10	–	10	–	ns
59	MCYC Delay Time	t_{MCYCD}	Fig. 6~ Fig. 10	–	60	–	50	–	50	ns
60	MRD Delay Time	t_{MRDD}	Fig. 6~ Fig. 9	–	90	–	80	–	70	ns
61	MRD Hold Time	t_{MRDH}		10	–	10	–	10	–	ns
62	$\overline{DRAW}$ Delay Time	t_{DRWD}		–	90	–	80	–	70	ns
63	$\overline{DRAW}$ Hold Time	t_{DRWH}		10	–	10	–	10	–	ns
64	Memory Write Data Delay Time	t_{MWDD}	Fig. 8	–	90	–	80	–	70	ns
65	Memory Write Data Hold Time	t_{MWDH}		10	–	10	–	10	–	ns
66										
67	$\overline{HSYNC}$ Delay Time	t_{HSD}	Fig. 9~ Fig. 11	–	90	–	80	–	70	ns
68	$\overline{VSYNC}$ Delay Time	t_{VSD}	Fig. 10	–	90	–	80	–	70	ns
69	$\overline{DISP_1}$, $\overline{DISP_2}$ Delay Time	t_{DSPD}		–	90	–	80	–	70	ns
70	$\overline{CUD_1}$, $\overline{CUD_2}$ Delay Time	t_{CUDD}		–	90	–	80	–	70	ns
71	$\overline{EXSYNC}$ Output Delay Time	t_{EXD}		20	90	20	80	20	70	ns
72	CHR Delay Time	t_{CHD}		–	90	–	80	–	70	ns
73										
74										
75	$\overline{EXSYNC}$ Input Pulse Width	t_{EXSW}	Fig. 11	3	–	3	–	3	–	Clk. Cyc.
76	$\overline{EXSYNC}$ Input Setup Time	t_{EXS}		60	–	60	–	50	–	ns
77	$\overline{EXSYNC}$ Input Hold Time	t_{EXH}		30	–	30	–	30	–	ns
78	LPSTB Uncertain Time 1	t_{LPD1}	Fig. 12	70	–	70	–	70	–	ns
79	LPSTB Uncertain Time 2	t_{LPD2}		10	–	10	–	10	–	ns
80	LPSTB Input Hold Time	t_{LPH}		10	–	10	–	10	–	ns
81	LPSTB Input Inhibit time	t_{LPI}		4	–	4	–	4	–	Clk. Cyc.

(to be continued)

No.	Item	Symbol	Measuring Condition	4 MHz Version HD63484-4		6 MHz Version HD63484-6		8 MHz Version HD63484-8		Unit
				min	max	min	max	min	max	
82	$\overline{DACK}$ Setup Time for $\overline{RES}$	t_{DAKSR}	Fig. 13	100	–	100	–	100	–	ns
83	$\overline{DACK}$ Hold Time for $\overline{RES}$	t_{DAKHR}		0	–	0	–	0	–	ns
84	$\overline{RES}$ Input Pulse Width	t_{RES}		10	–	10	–	10	–	Clk. Cyc.
85	$\overline{IRQ}$ Delay Time 1	t_{IRQ1}	Fig. 14	–	250	–	200	–	150	ns
86	$\overline{IRQ}$ Delay Time 2	t_{IRQ2}		–	500	–	500	–	500	ns
87	ATR Delay Time 1	t_{ATRD1}	Fig. 9	–	100	–	90	–	80	ns
88	ATR Hold Time 1	t_{ATRH1}		10	–	10	–	10	–	ns
89										
90	ATR Delay Time 2	t_{ATRD2}	Fig. 9	–	100	–	90	–	80	ns
91	ATR Hold Time 2	t_{ATRH2}		10	–	10	–	10	–	ns

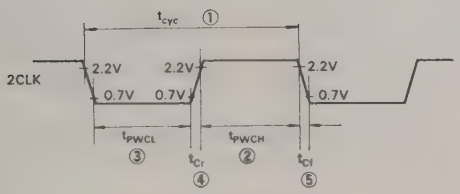


Figure 1 2CLK Waveform

Note: The timing specifications of the output waveform are defined at "High" level voltage = $V_{CC}-2.0V$, "Low" level voltage = 0.8V.

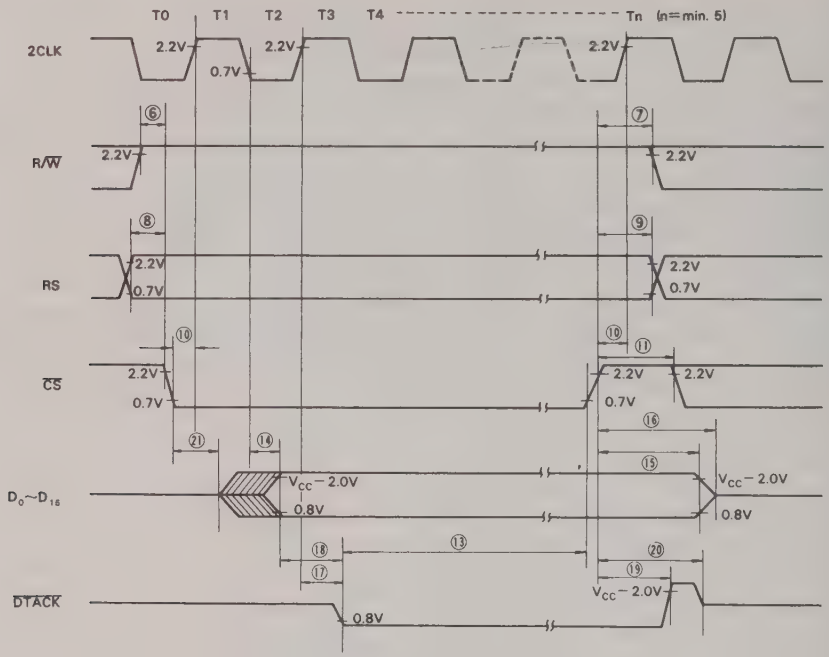


Figure 2 MPU Read Cycle Timing (MPU ← ACRTC)

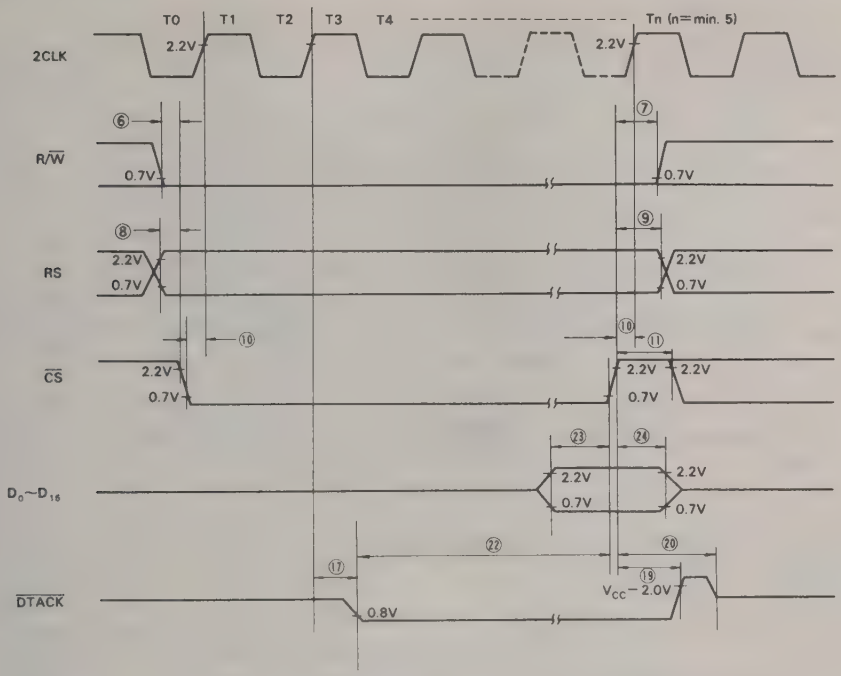


Figure 3 MPU Write Cycle Timing (MPU → ACRTC)

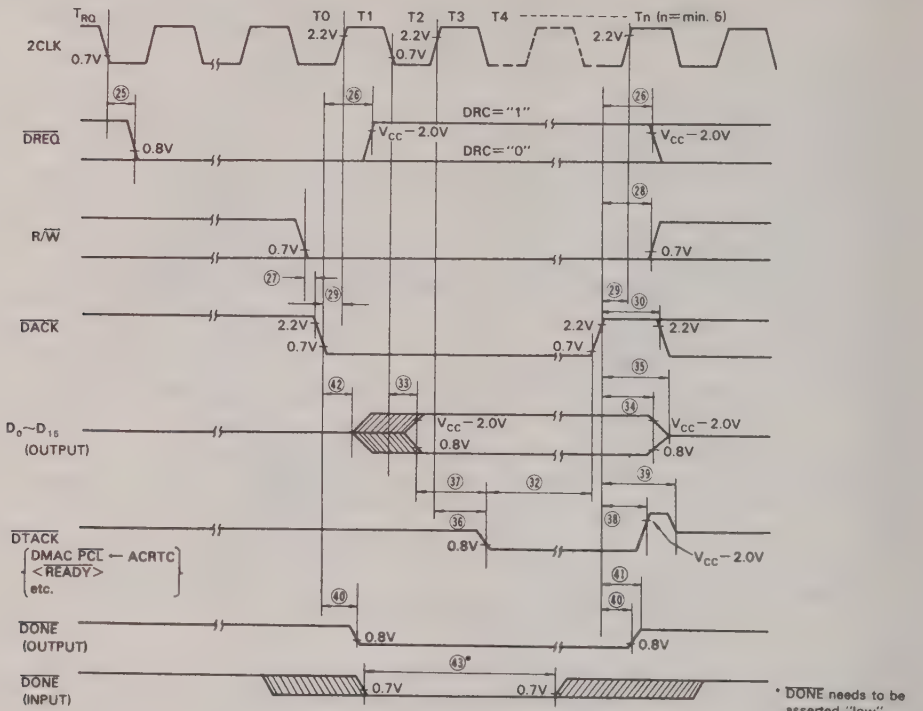


Figure 4 DMA Read Cycle Timing (Memory ← ACRTC)

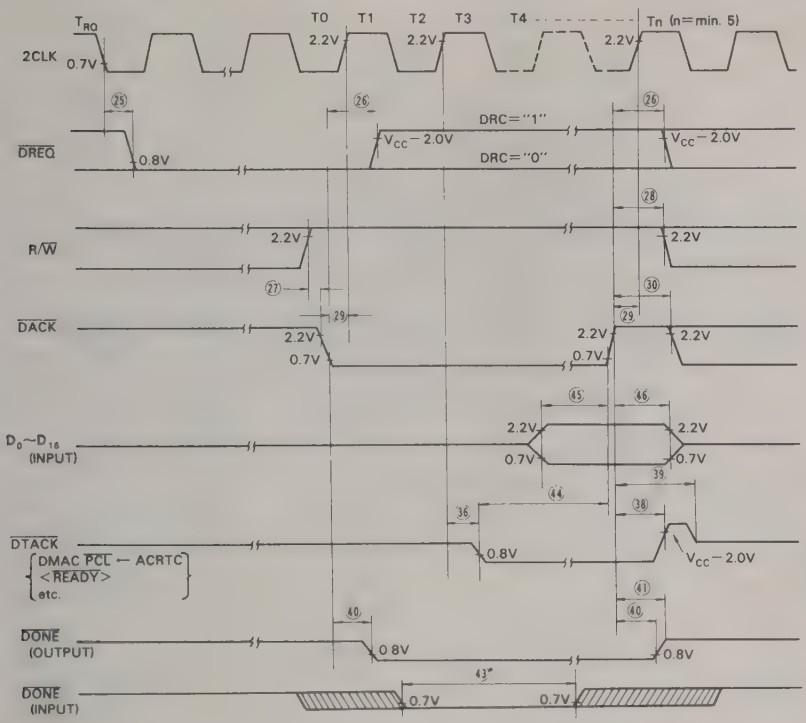


Figure 5 DMA Write Cycle Timing (Memory → ACRTC)

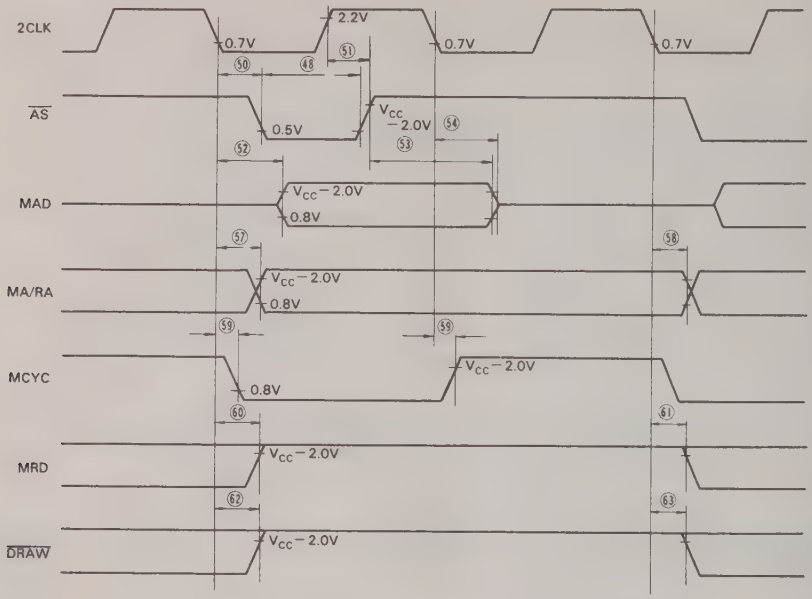


Figure 6 Screen Display Cycle Timing

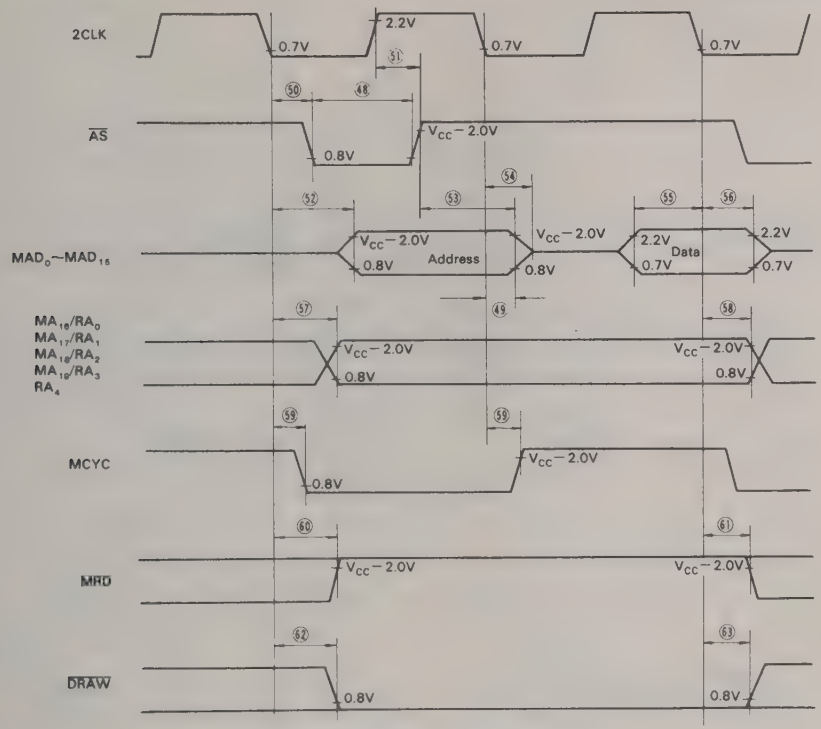


Figure 7 Frame Memory Read Cycle Timing
(ACRTC — Frame Memory)

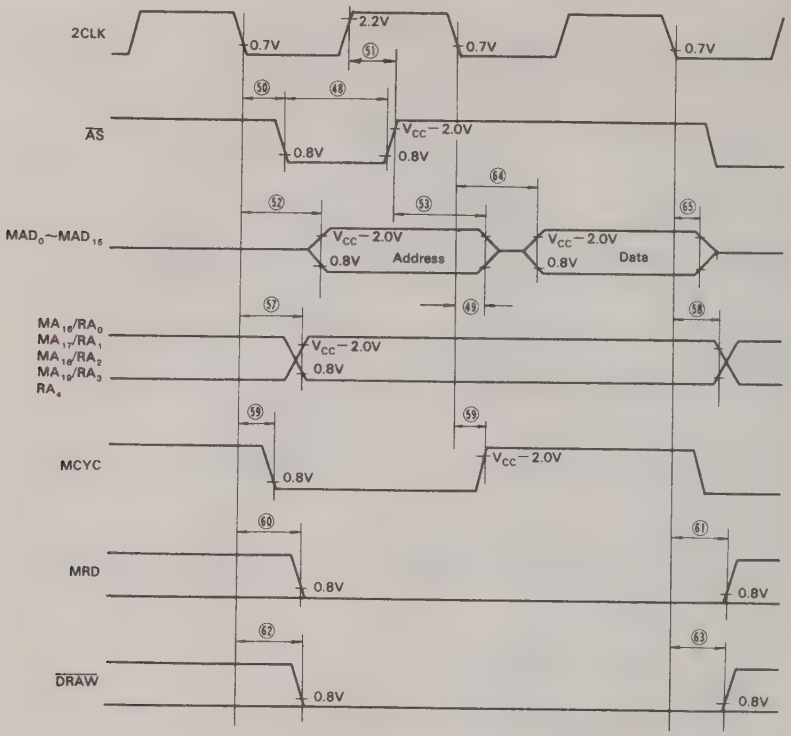


Figure 8 Frame Memory Write Cycle Timing (ACRTC → Frame Memory)

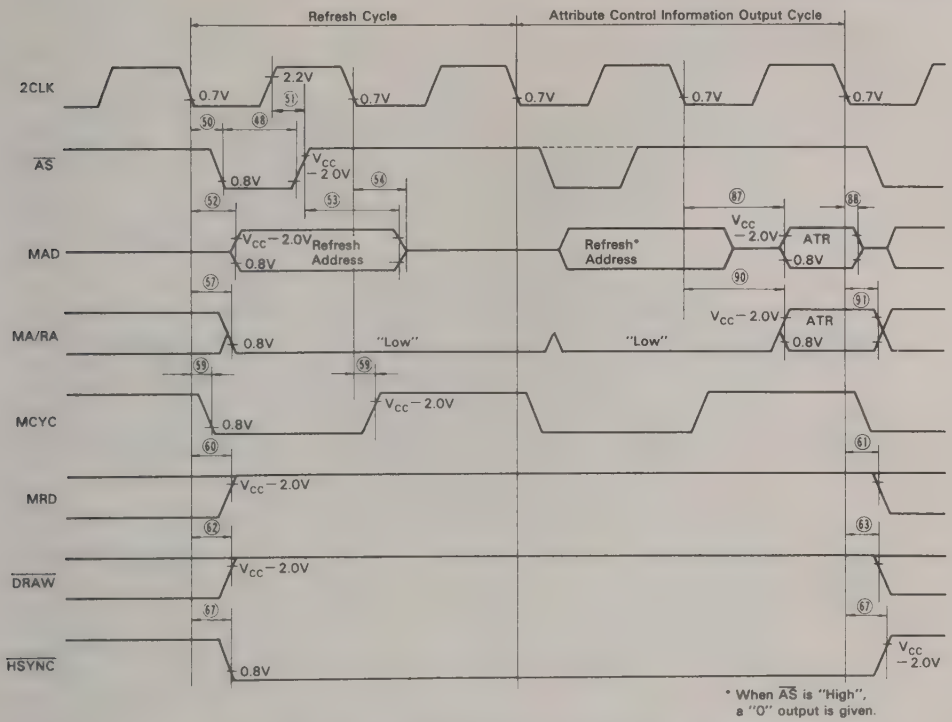


Figure 9 Frame Memory Refresh/Attribute Control Information Output Cycle Timing

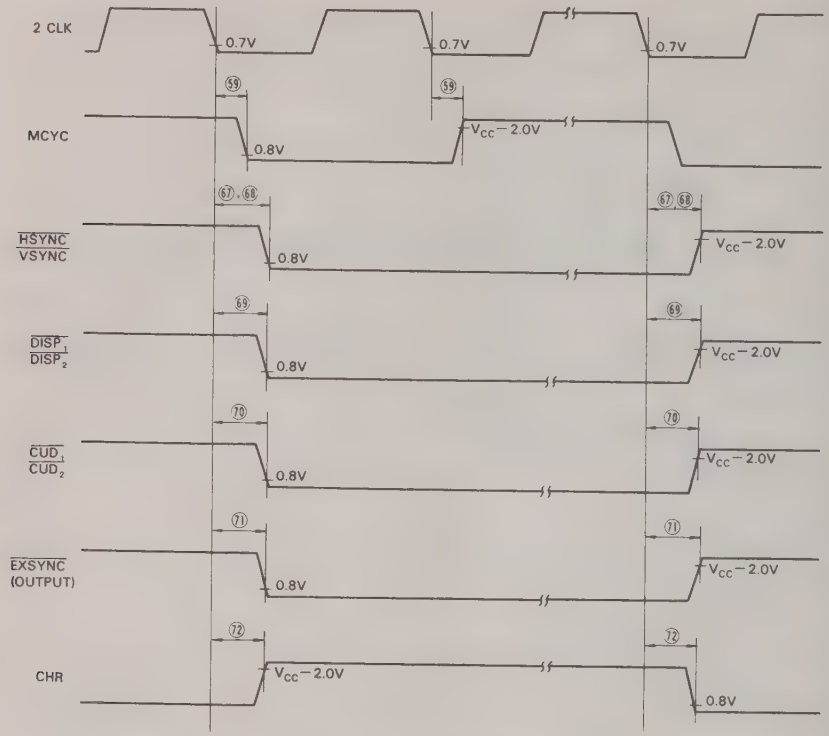


Figure 10 Display Control Signal Output Timing

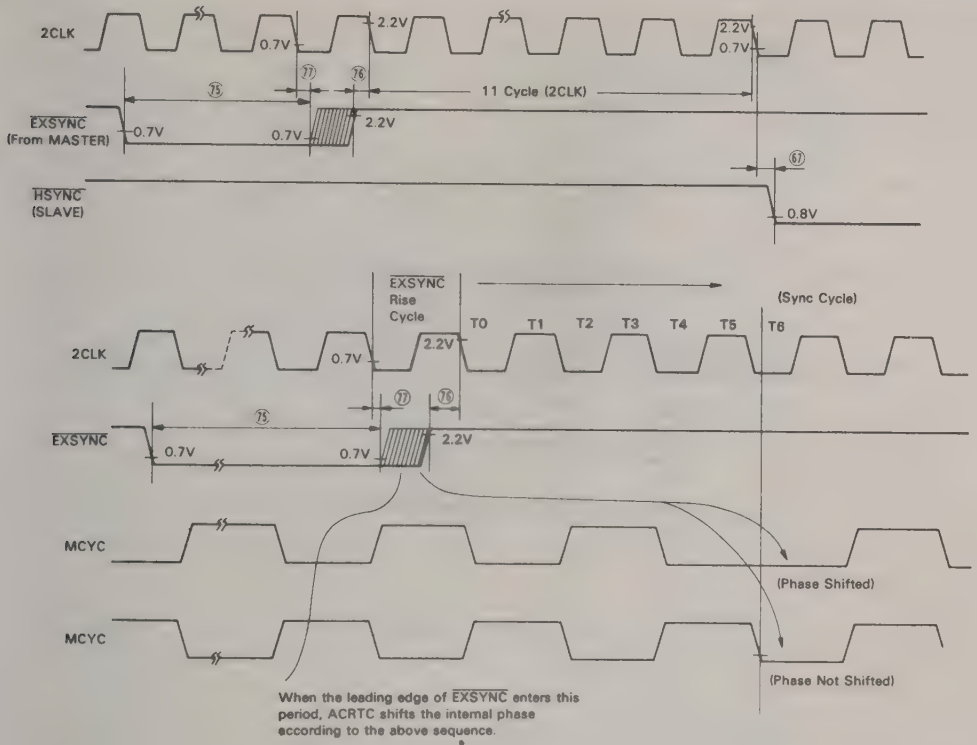


Figure 11 EXSYNC Input Timing

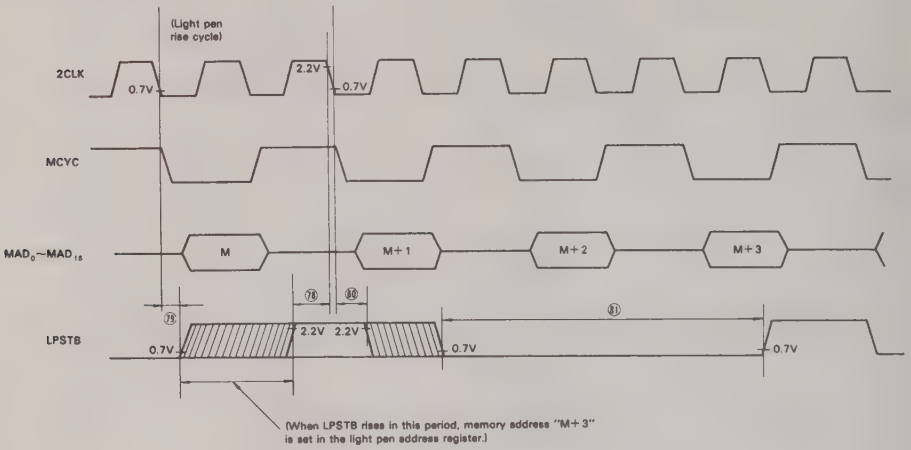


Figure 12 Input LPSTB Timing and Light Pen Address

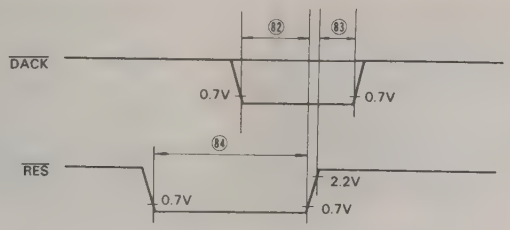


Figure 13 $\overline{RES}$ Input and $\overline{DACK}$ Input Timing
(System Reset and 16-bit/8-bit Selection)

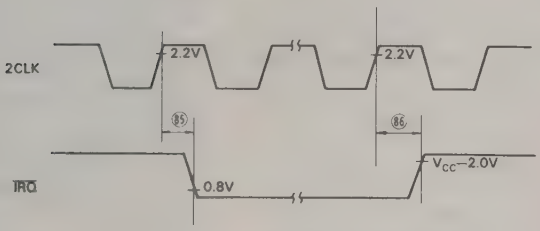
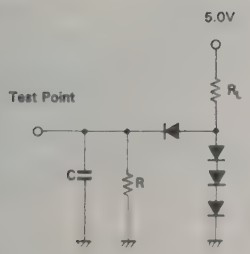


Figure 14 $\overline{IRQ}$ Output Timing



Signal	Load Condition
D_0—D_{15} $\overline{DTACK}$ $\overline{DREQ}$ MAD_0—MAD_{15} MA_{16}/RA_0—MA_{19}/RA_3 RA_4 $\overline{VSYNC}$, $\overline{HSYNC}$ $\overline{EXSYNC}$ MCYC, $\overline{AS}$, MRD $\overline{DRAW}$, CHR $\overline{DISP_1}$, $\overline{DISP_2}$ $\overline{CUD_1}$, $\overline{CUD_2}$	$R_L = 1.8k\Omega$ C = 40pF R = 10kΩ All diodes are 1S2074Ⓗ's or the equivalent.

Figure 15 Test Load Circuit A

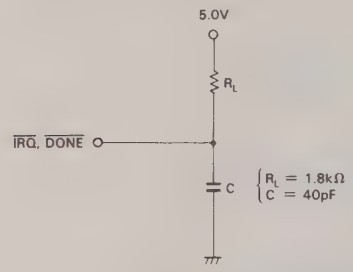
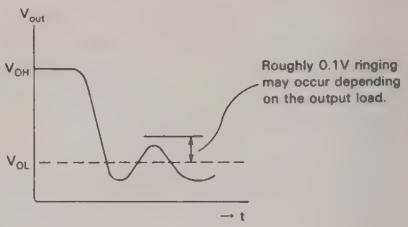


Figure 16 Test Load Circuit B



NOTE FOR USE

(1) Power-on Sequence
Following condition needs to be satisfied when the power turns-on.

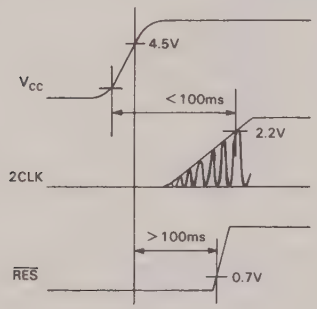
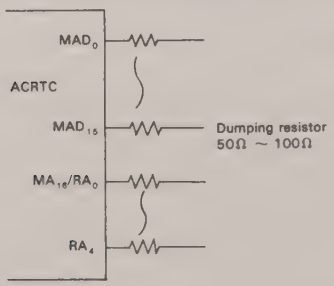


Figure 17 Power-on Sequence

(2) Output Waveform
Output ringing may occur on the CRT Address/Data Bus (MAD_0 ~ MAD_{15}, MA_{16}/RA_0 ~ MA_{19}/RA_3, RA_4) depending on the load of output line. Dumping resistors need to be connected when it could affect the correct operation in user's system.



■ ACRTC INTRODUCTION

• HD63484 (ACRTC) FEATURES

Powerful visual interfaces are a key component of advanced system architectures. A proven technique uses raster scanned CRT technology for the display of graphics and text information.

Systems which use first generation CRT Controllers (CRTCs) are constrained by hardware/software design time, manufacturing cost, and limited MPU bandwidth.

To meet the functional requirements for powerful visual interfaces, and to support their use in high volume, cost sensitive applications, advanced circuit design and VLSI CMOS manufacturing technologies have been used to create a next generation CRTC, the HD63484 ACRTC (Advanced CRT Controller).

The ACRTC concept is to incorporate major functionality, formerly requiring external hardware and software, on-chip. In this way, both higher performance and reduced system cost benefits are achieved.

* High Level Command Language Increases Performance and Reduces Software Development Cost.
  - ACRTC Converts Logical X-Y Coordinates to Physical Frame Buffer Addresses.
  - 38 Commands including 23 Graphic Drawing Commands – LINE, RECTANGLE, POLYLINE, POLYGON, CIRCLE, ELLIPSE, ARC, ELLIPSE ARC, FILLED RECTANGLE, PAINT, PATTERN and COPY.
  - On-chip 32 Byte Pattern RAM.
  - Conditional Drawing function (8 conditions) for Drawing Patterns, Color Mixing and Software Windowing.
  - Drawing Area Control with Hardware Clipping and Hitting.
  - Maximum Drawing Speed of 2 Million Logical Pixels per Second is the same for Monochrome and Color applications.
* High Resolution Display with Advanced Screen Control
  - Up to 4096 by 4096 Bit Map GRAPHIC Display and/or 256 Line by 256 Character by 32 Raster CHARACTER Display.
  - Separate Bit Map GRAPHIC (2M byte) and CHARACTER (128k byte) Address Spaces with Combined GRAPHIC/CHARACTER Display.
  - Three Horizontal Split Screens and One Window Screen. Size and Postition Fully Programmable.
  - Independent Horizontal and Vertical Smooth Scroll for each Screen.
  - 1 to 16 Zoom Magnitude – Independent X and Y Zoom Factors.
  - Logical Pixel Specification as 1, 2, 4, 8 or 16 Bits for Monochrome, Gray Scale and Color Displays.
  - Programmable Address Increment Supports Frame Buffer Memory Widths to 128 Bits for Video Bit Rates > 500 MHz.
  - Unique Interleaved Access Mode for Screen Superimposition or 'Flashless' Displays.
  - ACRTC provides Dynamic RAM Refresh Address.
* High Performance MPU Interface
  - Optimized Interface with the HD68000 MPU and HD68450 DMAC.

- 8 or 16 Bit Bus — Compatible With Other MPUs.
- Separate on-chip 16 Byte READ and WRITE FIFOs.
- Maskable Interrupts Including FIFO status.

• Versatile CRT Interface
- Full Programmability of CRT Timing Signals.
- Three Raster Scanning Modes.
- Master or Slave Synchronization to Multiple ACRTCs or Other Video Generating Devices.
- Two Hardware Cursors. Three Cursor Modes.
- Progreammable Cursor and Display Timing Skew.
- Eight User Defineable Video Attributes.
- Light Pen Detection.

• VLSI CMOS Process

• APPLICATIONS

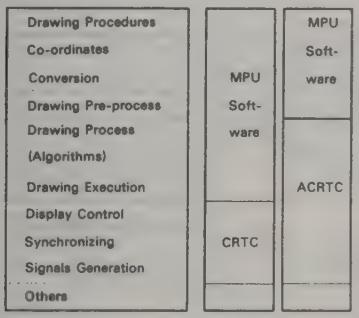


Figure 18 ACRTC vs. CRTC

The overall function of a visual interface is logically partitioned into layers. At the lowest layer are CRT timing and control signal generation. At the top layer are general purpose drawing procedures which provide a high-level interface to the users OS or application software. At this layer, a number of popular standards have emerged including GKS, CORE, NAPLP, GSX and others.

Figure 18 shows how the ACRTC performs the key functions or logical drawing algorithm and physical drawing execution. Formerly, these function were performed by external hardware and/or MPU software.

As shown, the ACRTC reduces the 'gap' between device functionality and high level graphics procedures. Since the ACRTC device itself provides capabilities closely related to those of high level graphics packages, the effort (hardware and software design time and cost) required to develop a visual interface is significantly reduced.

Noting the traditional and emerging applications for visual interfaces, figure 19 shows that a single ACRTC is suitable for a broad range of products in both alphanumeric and graphics areas.

Multiple ACRTCs can achieve performance beyond that of any first generation CRTC configuration.

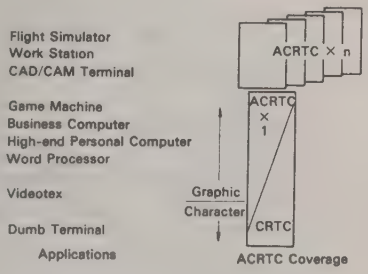


Figure 19 Application Spectrum

• SYSTEM CONFIGURATION

Existing CRTCs provide a single bus interface to the frame buffer which must be shared with the host MPU. However, the refresh of large frame buffers and the requirement to access the frame buffer for drawing operations can quickly saturate this shared bus bandwidth.

As shown, the ACRTC uses separate host MPU and frame buffer bus interfaces. This allows the ACRTC full access to the frame buffer for display refresh, DRAM refresh and drawing operations while minimizing the ACRTCs usage of the MPU system bus. Thus, overall system performance is maximized. A related benefit is that a large frame buffer (2M byte for each ACRTC) is useable even if the host MPU has a smaller address space or segment size restriction.

The ACRTC can utilize an external DMA Controller. This increases system throughput when large amounts of command, parameter and data information must be transferred to the ACRTC. Also, advanced DMAC features, such as the HD68450

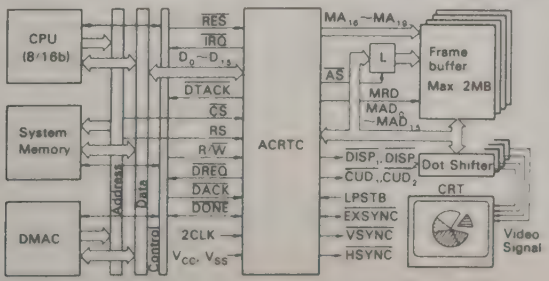


Figure 20 System Configuration

DMACs 'chaining' modes, can be used to develop powerful graphics system architectures.

However, more cost sensitive or less performance sensitive applications do not require a DMAC. The interface to the ACRTC can be handled completely under MPU software control.

While both ACRTC bus interfaces (Host MPU and Frame Buffer) exploit 16 bit data paths for maximum performance, the ACRTC also offers an 8 bit MPU mode for easy connection to popular 8 bit bus structures.

■ INTERNAL FUNCTIONS

• BLOCK DIAGRAM

The ACRTC consists of five major functional blocks. These functional blocks operate in parallel to achieve maximum performance. Two of the blocks perform the external bus interface for the host MPU and CRT respectively.

○ MPU Interface

Manages the asynchronous host MPU interface including the programmable interrupt control unit and DMA handshaking

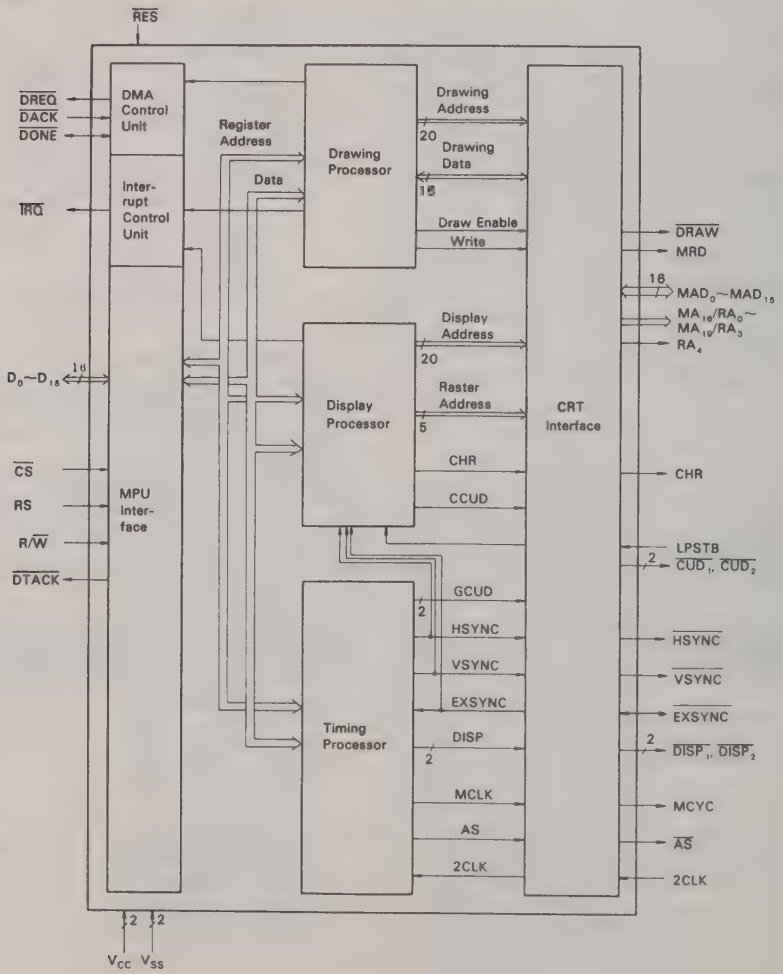


Figure 21 Block Diagram

control unit.

- CRT Interface
  Manages the frame buffer bus and CRT timing input and output control signals. Also, the selection of either display refresh address or drawing address outputs is performed.
  The other three blocks are separately microprogrammed processors which operate in parallel to perform the major functions of drawing, display control and timing.
- Drawing Processor
  Interprets commands and command parameters issued by the host bus (MPU and/or DMAC) and performs the drawing operations on the frame buffer memory. This processor is responsible for the execution of ACRTC drawing algorithms and conversion of logical pixel X-Y addresses to physical frame buffer addresses.
  Communication with the host bus is via separate 16 byte read and write FIFOs.
- Display Processor
  Manages frame buffer refresh addressing based on the user programmed specification of display screen organization. Combines and displays as many as 4 independent screen segments (3 horizontal splits and 1 window) using an internal high speed address calculation unit. Controls display refresh address outputs based on GRAPHIC (physical frame buffer address) or CHARACTER (physical frame buffer address + row address) display modes.
- Timing Processor
  Generates the CRT synchronization signals and other timing signals used internally by the ACRTC.
  The ACRTCs software visible registers are similarly partitioned and reside in the appropriate internal processor depending on function. The registers in the Display and Timing processors are loaded with basic display parameters during system initialization. During operation, the host primarily communicates with the ACRTCs Drawing processor via the on-chip FIFOs.

• SIGNAL DESCRIPTION

Following is a brief description of the ACRTC pin functions organized as MPU Interface, DMAC Interface, CRT Interface and Power Supply.

MPU INTERFACE

$\overline{RES}$ — Input
Hardware reset input to the ACRTC.

$D_0 \sim D_{15}$ — Input/Output
The bidirectional data bus for communication with the host MPU or DMAC. In 8 bit data bus mode, $D_0 \sim D_7$ are used.

R/$\overline{W}$ — Input
Controls the direction of host ↔ ACRTC transfers.

$\overline{CS}$ — Input
Enables data transfers between the host and the ACRTC.

RS — Input
Selects the ACRTC register to be accessed and is normally connected to the least significant bit of the host address bus.

$\overline{DTACK}$ — Output
Provides asynchronous bus cycle timing and is compatible with the HD68000 MPU $\overline{DTACK}$ input.

$\overline{IRQ}$ — Output
Generates interrupt service requests to the host MPU.

DMAC INTERFACE

$\overline{DRFQ}$ — Output
Generates DMA service requests to the host DMAC.

$\overline{DACK}$ — Input
Receives DMA acknowledge timing from the host DMAC.

$\overline{DONE}$ — Input/Output
Terminates DMA transfer and is compatible with the HD68450 DMAC $\overline{DONE}$ signal.

CRT INTERFACE

2CLK — Input
Basic ACRTC operating clock derived from the dot clock.

$MAD_0 \sim MAD_{15}$ — Input/Output
Multiplexed frame buffer address/data bus.

$\overline{AS}$ — Output
Address strobe for demultiplexing the frame buffer address/data bus ($MAD_0 \sim MAD_{15}$).

$MA_{16}/RA_0 \sim MA_{19}/RA_3$ — Output
The high order address bits for graphic screens and the raster address outputs for character screens.

RA_4 — Output
Provides the high order raster address bit (up to 32 rasters) for character screens.

CHR — Output
Indicates whether a graphic or character screen is being accessed.

MCYC — Output
Frame buffer memory access timing — one half the frequency of 2CLK.

MRD — Output
Frame Buffer data bus direction control.

$\overline{DRAW}$ — Output
Differentiates between drawing cycles and CRT display refresh cycles.

$\overline{DISP_1}$, $\overline{DISP_2}$ — Output
Programmable display enable timing used to selectively enable, disable and blank logical screens.

$\overline{CUD_1}$, $\overline{CUD_2}$ — Output
Provides cursor timing determined by ACRTC programmed parameters such as cursor definition, cursor mode, cursor address, etc.

$\overline{VSYNC}$ — Output
CRT device vertical synchronization pulse.

$\overline{HSYNC}$ — Output
CRT device horizontal synchronization pulse.

$\overline{EXSYNC}$ — Input/Output
For synchronization between multiple ACRTCs and other video signal generating devices.

LPSTB — Input
Connection to an external light pen.

VIDEO ATTRIBUTES

The ACRTC outputs 20 bits of video attributes on $MAD_0 \sim MAD_{15}$ and $MA_{16}/RA_0 \sim MA_{19}/RA_3$. These attributes are out-

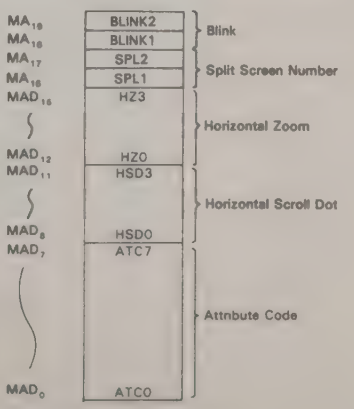


Figure 22 Video Attributes

put at the last cycle prior to the rising edge of $\overline{\text{HSYNC}}$ and should be latched externally. Thus, video attributes can be set on a raster by raster basis.

Attribute Code (ATC0~ATC7:$MAD_0 \sim MAD_7$)

These are user defined attributes. The programmed contents of the Attribute Control bits (ATR) of the Display Control Register (DCR) are output on these lines.

Horizontal Scroll Dot (HSD0~HSD3:$MAD_8 \sim MAD_{11}$)

These are used in conjunction with external circuitry to implement smooth horizontal scroll. These lines contain the encoded start dot address which is used to control the external shift register load timing and data. HSD usually corresponds to the start dot address of the background screens. However, if the window smooth scroll (SWS) bit of OMR (Operation Mode Register) is set to 1, HSD outputs the start dot address of the window screen segment.

Horizontal Zoom Factor (HZ0~HZ3:$MAD_{12} \sim MAD_{15}$)

These lines output the encoded (1-16) horizontal zoom factor as stored in the Zoom Factor Register (ZFR). Horizontal zoom is accomplished by the ACRTC repeating a single display address and using the HZ outputs to control the external shift register clock. Horizontal zoom can only be applied to the Base screen.

Split Position (SPL1~SPL2:$MA_{16} \sim MA_{17}$)

These lines present the encoded information showing the enabled background screen currently being displayed by the ACRTC.

SPL2	SPL1	
0	0	Background Screen not enabled or displayed
0	1	Base Screen
1	0	Upper Screen
1	1	Lower Screen

Blink (BLINK1~BLINK2:$MA_{18} \sim MA_{19}$)

The lines alternate from high to low periodically as defined in the Blink Control Register (BCR). the blink frequency is specified in units of 4 field times. A field is defined as the period between successive VSYNC pulses. These lines are used to implement character and screen blink.

• ADDRESS SPACE

The ACRTC allows the host to issue commands using logical X-Y coordinate addressing. The ACRTC converts these to physical linear word addresses with bit field offsets in the frame buffer.

Figure 23 shows the relationship between a logical X-Y screen address and the frame buffer memory, organized as sequential 16 bit words. The host may specify that a logical pixel consists of 1, 2, 4, 8 or 16 physical bits in the frame buffer. In the example, 4 bits per logical pixel is used allowing 16 colors or tones to be selected.

Up to four logical screens (Upper, Base, Lower and Window) are mapped into the ACRTC physical address space. The host specifies a logical screen physical start address, logical screen physical memory width (number of memory words per raster), logical pixel physical memory width (number of bits per pixel) and the logical origin physical address. Then, logical pixel X-Y addresses issued by the host or by the ACRTC Drawing processor are converted to physical frame buffer addresses. The ACRTC also performs bit extraction and masking to map logical pixel operations (in the example, 4 bits) to 16 bit word frame buffer accesses.

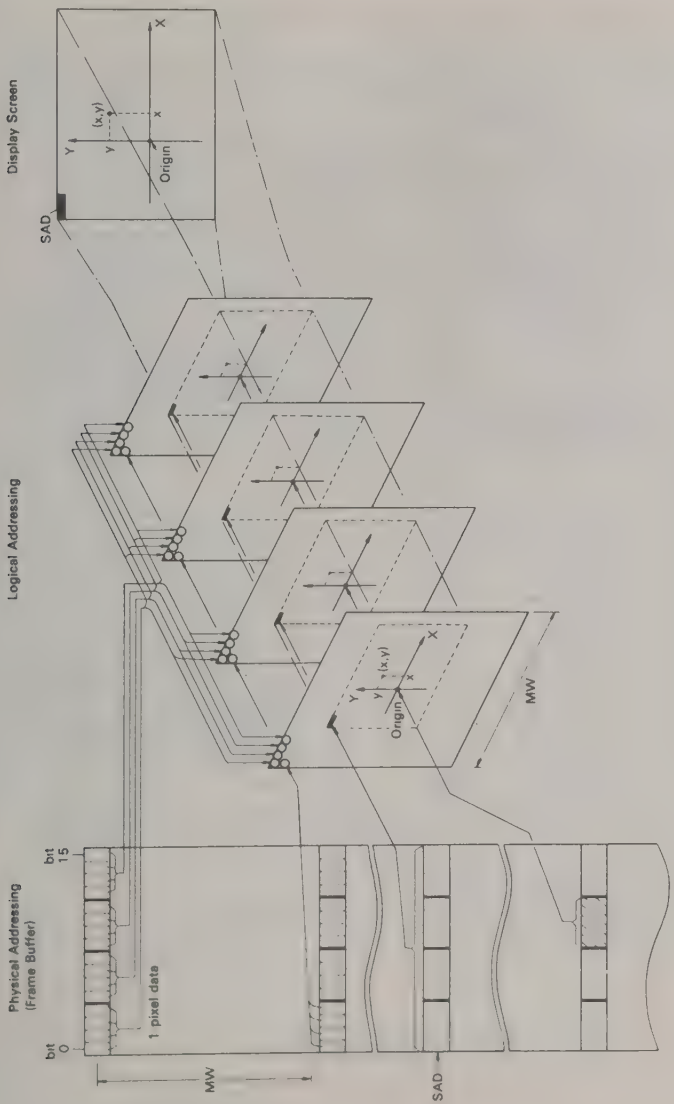


Figure 23 Logical/Physical Addressing

- **REGISTERS**

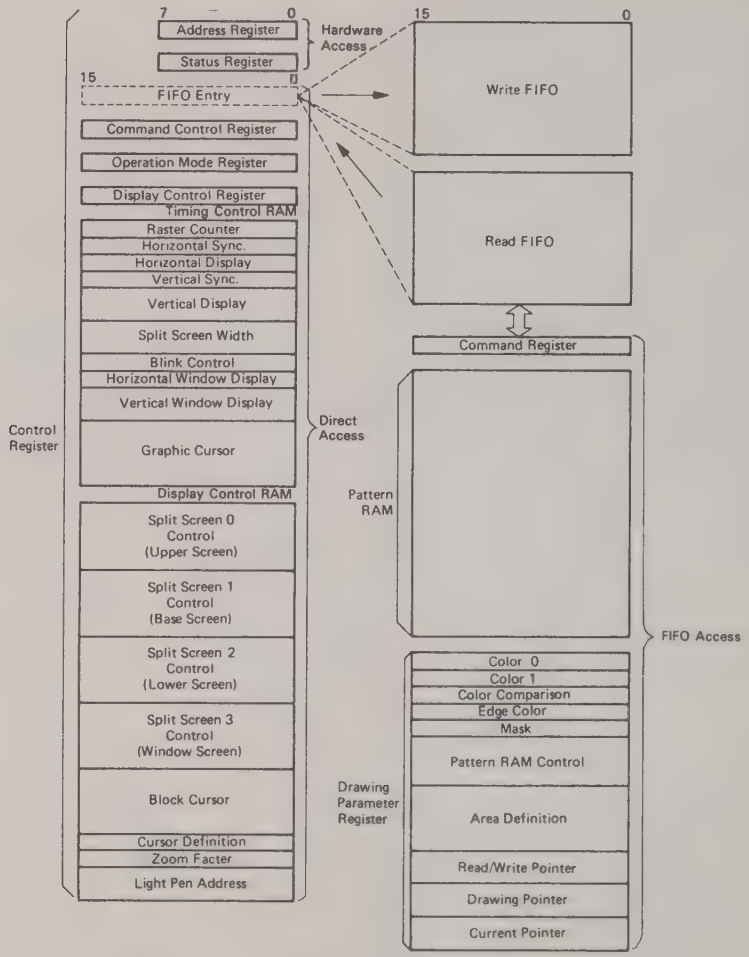


Figure 24 Programming Model

Table 1 Programming Model (Hardware Access, Direct Access Registers)

CS	RS	R/W	Reg. No.	Register Name	Abbr.	DATA (H) 15	14	13	12	11	10	9	8	DATA (L) 7	6	5	4	3	2	1	0
1	–	–	–	–						—							—				
0	0	0	AR	Address Register	AR					—							Address				
0	0	1	SR	Status Register	SR					—				CER	ARD	CED	LPD	RFF	RFR	WFR	WFE
		1/0	r00	FIFO Entry	FE									FE							
		1/0	r02	Command Control	CCR	ABT	PSE	DDM	CDM	DRC	GBM			CRE	ARE	CEE	LPE	RFE	RRE	WRE	WEE
		1/0	r04	Operation Mode	OMR	M/S	STR	ACP	WSS	CSK		DSK		RAM		GAI		ACM		RSM	
		1/0	r06	Display Control	DCR	DSP	SE1		SE0		SE2		SE3					ATR			
		–	r08 ↓ r7E			No USE and RESERVED															
		1	r80	Raster Count	RCR	—								RC							
		1/0	r82	Horizontal Sync.	HSR					HC				—				HSW			
		1/0	r84	Horizontal Display	HDR					HDS								HDW			
		1/0	r86	Vertical Sync.	VSR			—									VC				
		1/0	r88	Vertical Display	VDR					VDS				—				VSW			
0	1	1/0	r8A					—									SP1				
		1/0	r8C	Split Screen Width	SSW			—									SP0				
		1/0	r8E					—									SP2				
		1/0	r90	Blink Control	BCR			BON1				BOFF1					BON2			BOFF2	
		1/0	r92	Horizontal Window Display	HWR					HWS								HWW			
		1/0	r94	Vertical Window Display	VWR			—									VWS				
		1/0	r96					—									VWW				
		1/0	r98	Graphic Cursor	GCR			—		CXE								CXS			
		1/0	r9A					—									CYS				
		1/0	r9C					—									CYE				
		–	r9E	ACRTC Work Area	–					—							—				
		–	rA0 ↓ rBE	—		No USE and RESERVED															
		1/0	rC0	UPPER Raster Address 0	RAR0	—				LRA0				—				FRA0			
		1/0	rC2	(Back Memory Width 0	MWR0	CHR		—									MW0				
		1/0	rC4	Ground) Start Address 0	SAR0			—			SDA0						—			SA0H/SAR0	
		1/0	rC6											SA0L							
		1/0	rC8	BASE Raster Address 1	RAR1	—				LRA1				—				FRA1			
		1/0	rCA	(Back Memory Width 1	MWR1	CHR		—									MW1				
		1/0	rCC	Ground) Start Address 1	SAR1			—			SDA1						—			SA1H/SRA1	
		1/0	rCE											SA1L							
		1/0	rD0	LOWER Raster Address 2	RAR2	—				LRA2				—				FRA2			
		1/0	rD2	(back Memory Width 2	MWR2	CHR		—									MW2				
		1/0	rD4	Ground) Start Address 2	SAR2			—			SDA2									SA2H/SRA2	
		1/0	rD6	Star 3										SA2L							
		1/0	rD8	Window Raster Address 3	RAR3	—				LRA3				—				FRA3			
		1/0	rDA	Memory Width 3	MWR3	CHR		—									MW3				
		1/0	rDC	Start Address 3	SAR3			—			SDA3						—			SA3H/SRA3	
		1/0	rDE											SA3L							
		1/0	rE0	Block Cursor 1	BCUR1		BCW1				BCSR1			—				BCER1			
		1/0	rE2											BCA1							
		1/0	rE4	Block Cursor 2	BCUR2		BCW2				BCSR2			—				BCER2			
		1/0	rE6											BCA2							
		1/0	rE8	Cursor Definition	CDR		CM		CON1			COFF1		—		CON2			COFF2		
		1/0	rEA	Zoom Factor	ZFR		HZF				VZF			—							
		1	rEC	Light Pen Address	LPAR	—				—				CHR		—			LPAH		
		1	rEE											LPAL							
		–	rF0 ↓ rFE	—	–	No USE and RESERVED															

Note { 1 ... "High" level
0 ... "Low" level

HARDWARE ACCESS, DIRECT ACCESS REGISTER

ABT : Abort
ACM : Access Mode
ACP : Access Priority
Address : Register No. of the control register
ARD : Area Detect
ARE : Area Detect Interrupt Enable
ATR : Attribute Control
CDM : Command DMA Mode
CED : Command End
CEE : Commad End Interrupt Enable
CER : Command Error
CRE : Command Error Interrupt Enable
CSK : Cursor Display Skew
DDM : Data DMA Mode
DRC : DMA Request Control
DSK : $\overline{DISP}$ Skew
DSP : $\overline{DISP}$ Signal Control
FE : FIFO Entry
GAI : Graphic Address Increment Mode
GBM : Graphic Bit Mode
HC : Horizontal Cycle
HDS : Horizontal Display Start
HDW : Horizontal Display Width
HSW : Horizontal Sync. Width
LPD : Light Pen Strobe Detect
LPE : Light Pen Strobe Interrupt Enable
M/S : Master/Slave
PSE : Pause
RAM : RAM Mode
RC : Raster Count
RFE : Read FIFO Full Interrupt Enable
RFF : Read FIFO Full
RFR : Read FIFO Ready
RRE : Read FIFO Ready Interrupt Enable
RSM : Raster Scan Mode
SE0 : Split Enable 0
SE1 : Split Enable 1
SE2 : Split Enable 2
SE3 : Split Enable 3
STR : Start
VC : Vertical Cycle
VDS : Vertical Display Start
VSW : Vertical Sync. Width
WEE : Write FIFO Empty Interrupt Enable
WFE : Write FIFO Empty
WFR : Write FIFO Ready
WRE : Write FIFO Ready Interrupt Enable
WSS : Window Smooth Scroll

SP0, SP1, SP2 : Split Screen 0 Width, Split Screen 1 Width, Split Screen 2 Width
BON1, BON2 : Blink ON 1, Blink ON 2
BOFF1, BOFF2 : Blink OFF 1, Blink OFF 2
HWS : Horizontal Window Start
HWW : Horizontal Window Width
VWS : Vertical Window Start
VWW : Vertical Window Width
CXS, CYS : Cursor X Start, Cursor Y Start
CXE, CYE : Cursor X End, Cursor Y End
FRA : First Raster Address
LRA : Last Raster Address
CHR : Character
MW : Memory Width
SDA : Start Dot Address
SAH/SRA : Start Address "High"/Start Raster Address
SAL : Start Address "Low"
BCW1, BCW2 : Block Cursor Width 1, Block Cursor Width 2
BCSR1, BCSR2 : Block Cursor Start Raster 1, Block Cursor Start Raster 2
BCER1, BCER2 : Block Cursor End Raster 1, Block Cursor End Raster 2
BCA1, BCA2 : Block Cursor Address 1, Block Cursor Address 2
CM : Cursor Mode
CON1, CON2 : Cursor ON 1, Cursor ON 2
COFF1, COFF2 : Cursor OFF 1, Cursor OFF 2
HZF, VZF : Horizontal Zoom Factor, Vertical Zoom Factor
LPAH : Light Pen Address "High"
LPAL : Light Pen Address "Low"

Table 1 (cont.) Programming Model (Drawing Parameter Registers)

Register No	Read/ Write	Name of Register	Abbr.	Data (H)								Data (L)							
				15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Pr00	R/W	Color 0	CL0	CL0															
Pr01	R/W	Color 1	CL1	CL1															
Pr02	R/W	Color Comparison	CCMP	CCMP															
Pr03	R/W	Edge Color	EDG	EDG															
Pr04	R/W	Mask	MASK	MASK															
Pr05	R/W	Pattern RAM Control	PRC	PPY								PZCY				PPX			PZCX
↓				PSY												PSX			
Pr07				PEY								PZY				PEX			PZX
Pr08	R/W	Area Definition **	ADR	XMIN															
↓				YMIN															
				XMAX															
Pr0B				YMAX															
Pr0C	R/W	Read Write Pointer	RWP	DN								RWPH							
Pr0D				RWPL															
Pr0E	–	. . . . .	–	–															
Pr0F				–															
Pr10	R	Drawing Pointer	DP	DN								DPAH							
Pr11				DPAL														DPD	
Pr12	R	Current Pointer **	CP	X															
Pr13				Y															
Pr14	–	. . . . .	–	–															
Pr15				–															

* R . . . Register readable by a Read Parameter Register (RPR) command
W . . Register writable by a Write Parameter Register (WPR) command
– . . Access is not allowed
▨ . . . Always set to "0"
** . . Set binary complements for negative values of X and Y axis.

DRAWING PARAMETER REGISTER

R	: Register which can be read by Read Parameter Register Command (RPR)
W	: Register which can be written into by Write Parameter Register Command (WPR)
-	: Access is not allowed
CL0	: Defines the color data used for the drawing when logical drawing data = 0
CL1	: Defines the color data used for the drawing when logical drawing data = 1
CCMP	: Defines the comparative color of the drawing operation
EDG	: Defines the edge color
MASK	: Defines the bit pattern used to mask bits upon which data transfer should not be performed
PSX, PSY	: Pattern Start Point
PEX, PEY	: Pattern End Point
PPX, PPY	: Pattern Scan Start Point
PZX, PZY	: Pattern Zoom
PZCX, PZCY	: Pattern Zoom Count
XMIN, YMIN	: Start point of Area definition
XMAX, YMAX	: End point of Area definition
DN	: Screen Number
RWPH	: High-order 8 bit of Read Write Pointer Address
RWPL	: Low-order 12 bit of Read Write Pointer Address
DPAH	: High-order 8 bit of Drawing Pointer Address
DPAL	: Low-order 12 bit of Drawing Pointer Address
DPD	: Drawing Pointer Dot Address
X, Y	: Position indicated by Current Pointer on X-Y coordinate

The ACRTC has over two hundred bytes of accessible registers. These are organized as Hardware, Directly and FIFO accessible.

- Hardware Accessible

  The ACRTC is connected to the host MPU as a standard peripheral which occupies two word locations of the host address space. The RS (Register Select) pin selects one of these two locations. When RS is low, reads access the Status Register and writes access the Address Register.

  The Status Register summarizes the ACRTC state and is used by the MPU to monitor the overall operation of the ACRTC.

  The Address Register is used to program the ACRTC with the address of the specific directly accessible register which the MPU wishes to access.

- Directly Accessible

  These registers are accessed by prior loading of the Address Register with the chosen register address. Then, when the MPU accesses the ACRTC with RS=1, the chosen register is accessed.

  The FIFO entry enables access to FIFO accessible registers using the ACRTC read and write FIFOs.

  The Command Control Register is used to control overall ACRTC operation such as aborting or pausing commands, defining DMA protocols, enabling/disabling interrupt sources, etc.

  The Operation Mode Register defines basic parameters of ACRTC operation such as frame buffer access mode, display or drawing priority, cursor and display timing skew factors, raster scan mode, etc.

  The Display Control Register allows the independent enabling and disabling of each of the four ACRTC logical display screens (Base, Upper, Lower and Window). Also, this register contains the 8 bits of user defineable video attributes.

  The Timing Control RAM contains registers which define ACRTC timing. This includes timing specification for CRT control signals (e.g. $\overline{\text{HSYNC}}$, $\overline{\text{VSYNC}}$), logical display screen size and display period, blink timing, etc.

  The Display Control RAM contains registers which define logical screen display parameters such as start addresses, raster addresses and memory width. Also included are the cursor(s) definition, zoom factor and light pen registers.

- FIFO Accessible

  For high performance drawing, key Drawing Processor registers are coupled to the host via the ACRTCs separate 16 byte read and write FIFOs.

  ACRTC commands are sent from the MPU via the write FIFO to the Command register. As the ACRTC completes command execution, the next command is automatically fetched from the FIFO into the Command register.

  The Pattern RAM is used to define drawing and painting 'patterns'. The Pattern RAM is accessed using the ACRTCs Read Pattern RAM (RPTN) and Write Pattern RAM (WPTN) register access commands.

  The Drawing Parameter Registers define detailed parameters of the drawing process, such as color control, area control (hitting/clipping) and Pattern RAM pointers. The Drawing Parameter Registers are accessed using the ACRTCs Read Parameter Register (RPR) and Write Parameter Register (WPR) register access commands.

■ COMMANDS

The ACRTC has 38 commands classified into three groups — REGISTER ACCESS, DATA TRANSFER and GRAPHIC DRAWING.

Five REGISTER ACCESS commands allow access to Drawing processor Drawing Parameter Registers and the Pattern RAM.

Ten DATA TRANSFER commands are used to move data between the host system memory and the frame buffer, or within the frame buffer.

Twenty three GRAPHIC DRAWING commands cause the ACRTC to perform drawing operations. Parameters for these commands are specified using logical X-Y addressing.

All the above commands, parameters and data are transferred via the ACRTC read and write FIFOs.

Assuming the ACRTC has been properly initialized, the MPU must perform two steps to cause graphic drawing.

First, the MPU must specify certain drawing parameters which define a number of details associated with the drawing process. For excample, to draw a figure or paint an area, the MPU must specify the drawing or painting 'pattern' by initializing the ACRTC Pattern RAM and related pointers. Also, if clipping and hitting control are desired, the MPU specifies the 'arera' to be monitored during drawing by initializing area definition registers. Other drawing parameters include color, edge definition, etc.

After the drawing parameters have been specified, the MPU issues a graphic drawing command and any required command parameters, such as the CRCL (Circle) command with a radius parameter. The ACRTC then performs the specified drawing operation by reading, modifying and rewriting the contents of the frame buffer.

Table 2 ACRTC Command Table

TYPE	MNEMONIC	COMMAND NAME	OPERATION CODE	PARAMETER	# (words)	~ (cycles)
Register Access Command	ORG	Origin	0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0	DPH DPL	3	8
	WPR	Write Parameter Register	0 0 0 0 1 0 0 0 0 0 0 RN	D	2	6
	RPR	Read Parameter Register	0 0 0 0 1 1 0 0 0 0 0 RN		1	6
	WPTN	Write Pattern RAM	0 0 0 1 1 0 0 0 0 0 0 PRA	n D1, ..., Dn	n+2	4n+8
	RPTN	Read Pattern RAM	0 0 0 1 1 1 0 0 0 0 0 PRA	n	2	4n+10
Data Transfer Command	DRD	DMA Read	0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0	AX AY	3	(4x+8)y+12⌈x·y/8⌉+(62~68)
	DWT	DMA Write	0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0	AX AY	3	(4x+8)y+16⌈x·y/8⌉+34
	DMOD	DMA Modify	0 0 1 0 1 1 0 0 0 0 0 0 0 MM	AX AY	3	(4x+8)y+16⌈x·y/8⌉+34
	RD	Read	0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0		1	12
	WT	Write	0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0	D	2	8
	MOD	Modify	0 1 0 0 1 1 0 0 0 0 0 0 0 MM	D	2	8
	CLR	Clear	0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0	D AX AY	4	(2x+8)y+12
	SCLR	Selective Clear	0 1 0 1 1 1 0 0 0 0 0 0 0 MM	D AX AY	4	(4x+8)y+12
	CPY	Copy	0 1 1 0 S DSD 0 0 0 0 0 0 0 0	SAH SAL AX AY	5	(6x+10)y+12
	SCPY	Selective Copy	0 1 1 1 S DSD 0 0 0 0 0 0 MM	SAH SAL AX AY	5	(6x+10)y+12
Graphic Command	AMOVE	Absolute Move	1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	X Y	3	56
	RMOVE	Relative Move	1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0	dX dY	3	56
	ALINE	Absolute Line	1 0 0 0 1 0 0 0 AREA COL OPM	X Y	3	P·L+18
	RLINE	Relative Line	1 0 0 0 1 1 0 0 AREA COL OPM	dX dY	3	P·L+18
	ARCT	Absolute Rectangle	1 0 0 1 0 0 0 0 AREA COL OPM	X Y	3	2P(A+B)+54
	RRCT	Relative Rectangle	1 0 0 1 0 1 0 0 AREA COL OPM	dX dY	3	2P(A+B)+54
	APLL	Absolute Polyline	1 0 0 1 1 0 0 0 AREA COL OPM	n X1, Y1, ... Xn, Yn	2n+2	Σ[P·L+16]+8
	RPLL	Relative Polyline	1 0 0 1 1 1 0 0 AREA COL OPM	n dX1, dY1, ... dXn, dYn	2n+2	Σ[P·L+16]+8
	APLG	Absolute Polygon	1 0 1 0 0 0 0 0 AREA COL OPM	n X1, Y1, ... Xn, Yn	2n+2	Σ[P·L+16]+P·Lo+20
	RPLC	Relative Polygon	1 0 1 0 0 1 0 0 AREA COL OPM	n dX1, dY1, ... dXn, dYn	2n+2	Σ[P·L+16]+P·Lo+20
	CRCL	Circle	1 0 1 0 1 0 0 C AREA COL OPM	r	2	8d+66
	ELPS	Ellipse	1 0 1 0 1 1 0 C AREA COL OPM	a b dX	4	10d+90
	AARC	Absolute Arc	1 0 1 1 0 0 0 C AREA COL OPM	Xc Yc Xe Ye	5	8d+18
	RARC	Relative Arc	1 0 1 1 0 1 0 C AREA COL OPM	dXc dYc dXe dYe	5	8d+18
	AEARC	Absolute Ellipse Arc	1 0 1 1 1 0 0 C AREA COL OPM	a b Xc Yc Xe Ye	7	10d+96
	REARC	Relative Ellipse Arc	1 0 1 1 1 1 0 C AREA COL OPM	a b dXc dYc dXe dYe	7	10d+96
	AFRCT	Absolute Filled Rectangle	1 1 0 0 0 0 0 0 AREA COL OPM	X Y	3	(P·A+B)B+18
	RFRCT	Relative Filled Rectangle	1 1 0 0 0 1 0 0 AREA COL OPM	dX dY	3	(P·A+B)B+18
	PAINT	Paint	1 1 0 0 1 0 0 E AREA 0 0 0 0 0		1	(18A+102)B-58 *1)
	DOT	Dot	1 1 0 0 1 1 0 0 AREA COL OPM		1	8
	PTN	Pattern	1 1 0 1 SL SD AREA COL OPM	SZ *2)	2	(P·A+10)B+20
	AGCPY	Absolute Graphic Copy	1 1 1 0 S DSD AREA 0 0 OPM	Xs Ys DX DY	5	((P+2)A+10)B+70
	RGCPY	Relative Graphic Copy	1 1 1 1 S DSD AREA 0 0 OPM	dXs dYs DX DY	5	((P+2)A+10)B+70

*1) In case of rectangular filling

*2) SZ: 16 | SZy | 87 | SZx | 0 — SZy, SZx: Pattern Size

REGISTER ACCESS COMMAND

RN : Register number of the drawing parameter register ($0-$13)
PRA : Pattern RAM address at which read write operation starts
DPH : Drawing pointer register High word
DPL : Drawing pointer register Low word

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DPH	DN		0	0	0	0	0	0	DPAH							
DPL	DPAL														DPD	

DPAH : Higher 8 bits of Drawing Pointer address
DPAL : Lower 12 bits of Drawing Pointer address
DPD : Dot position in the memory address

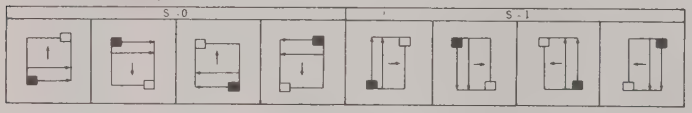
D, D_1,·····, Dn : Write data
n : Number of read write data

DN	Screen No.
00	Upper Screen
01	Base Screen
10	Lower Screen
11	Window Screen

DATA TRANSFER COMMAND

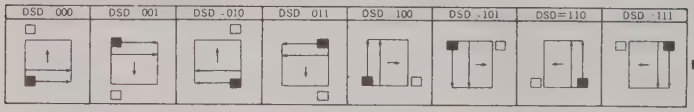
MM : Modify Mode

S : Source Scan Direction



■ : source area start address
□ : source area end address

DSD : Destination Scan Direction



■ : Read Write pointer start point
□ : Read Write pointer end point

AX : Number of word in X-axis direction − 1
AY : Number of word in Y-axis direction − 1
D : Write data
SAH : Source Start Address High word
SAL : Source Start Address Low word

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SAH	0	0	0	0	0	0	0	0	(SAH)							
SAL	(SAL)												0	0	0	0

(SAH) : Memory address Higher 8 bits
(SAL) : Memory address Lower 12 bits

x : Number of word in X-axis direction
y : Number of word in Y-axis direction
↑ : Rounding up

GRAPHIC DRAWING COMMAND

AREA : Area Mode
COL : Color Mode
OPM : Operation Mode

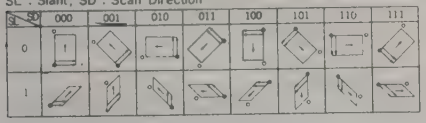
C : Circling Direction

C	Direction
0	Counterclockwise
1	Clockwise

E : Definition of Edge color

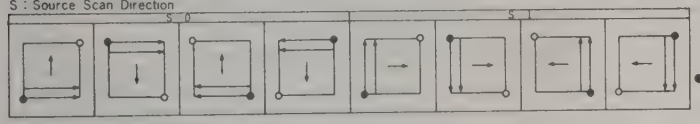
E	Definition
0	Edge color is defined by the data in the edge color register
1	Edge color is defined by the data excluding the above.

SL : Slant, SD : Scan Direction



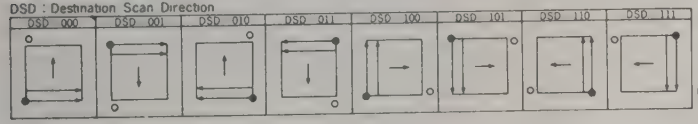
● : current pointer start point
: current pointer end point

S : Source Scan Direction



● : source area start dot position
: source area end dot position

DSD : Destination Scan Direction



● : current pointer start point
: current pointer end point

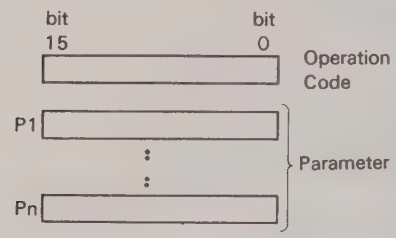
X, X1, ..., Xn : Absolute X-address from the origin point
Y, Y1, ..., Yn : Absolute Y-address from the origin point
dX : Relative X-address from the current pointer
dY : Relative Y-address from the current pointer
n : Number of nodes
dX1, ..., dXn : Relative X-address from each node
dY1, ..., dYn : Relative Y-address from each node
r : Dot number on radius
a, b : $(DX)^2 : (DY)^2 = a : b$
DX : X-direction dot number
DY : Y-direction dot number
Xc : Absolute X-address of the center point of arc/ellipse
Yc : Absolute Y-address of the center point of arc/ellipse
dXc : Relative X-address from the current pointer to the center point of arc/ellipse
dYc : Relative Y-address from the current pointer to the center point of arc/ellipse
Xe : Absolute X-address of the end point of arc/ellipse
Ye : Absolute Y-address of the end point of arc/ellipse
dXe : Relative X-address from the current pointer to the end point of arc/ellipse
dYe : Relative Y-address from the current pointer to the end point of arc/ellipse
Xs : Absolute X-address of the start dot position in source area
Ys : Absolute Y-address of the start dot position in source area
dXs : Relative X-address from the current pointer to the start dot position in source area
dYs : Relative Y-address from the current pointer to the start dot position in source area
P : 4(OPM=000~011)/6(OPM=100~111)
L, L0 : Dot number on straight line
d : total dot number
A : Scan main direction dot number
B : Scan sub direction dot number

- **COMMAND FORMAT**

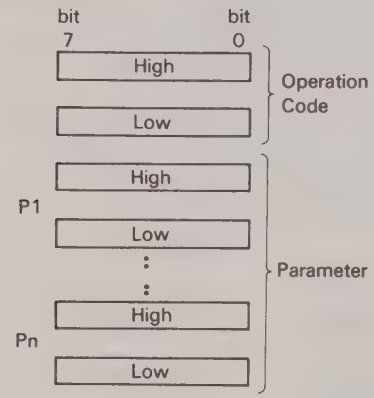
ACRTC commands consist of a 16 bit op-code, optionally followed by 1 or more 16 bit parameters. When 8 bit MPU mode is used, commands, parameters and data are sent to and from the ACRTC in the order of high byte, low byte.

(a) 16 bit interface

In the case of 16 bit interface, first move the 16 bit operation code and then move necessary 16 bit parameters one by one.



(b) 8 bit interface



In the case of 8 bit interface, first move the operation code's High byte and Low byte in this order and then move those of parameters in the same order.

- **COMMAND TRANSFER MODES**

Commands (and associated parameters) can be issued to the ACRTC in one of two ways — program transfer or DMA transfer.

PROGRAM TRANSFER

Program Transfer occurs when the MPU specifies the FIFO entry address and then writes commands/parameters to the write FIFO under program control (RS = high, R/$\overline{W}$, $\overline{CS}$ = low). The MPU writes are normally synchronized with ACRTC FIFO status by software polling or interrupts.

COMMAND DMA TRANSFER

Commands and parameters can be transferred from MPU system memory using in external DMAC. The MPU initiates and terminates Command DMA Transfer mode under software control (CDM bit of CCR). Command DMA can also be terminated by assertion of the ACRTC $\overline{DONE}$ signal. $\overline{DONE}$ is treated as an input in Command DMA Transfer Mode.

Using Command DMA Transfer, the ACRTC will issue cycle stealing DMA requests to the DMAC when the write FIFO is empty. The DMA data is automatically sent from system memory to the ACRTC write FIFO regardless of the contents of the Address Register.

- **REGISTER ACCESS COMMANDS**

Registers associated with the Drawing processor (the Pattern RAM and Drawing Parameter Registers) are accessed through the read and write FIFOs using the Register Access Commands.

- **DATA TRANSFER COMMANDS**

Data Transfer Commands are used to move blocks of data between the MPU system memory and the ACRTC frame buffer or within the frame buffer itself. Before issuing these commands, a physical 20 bit frame buffer address must be specified in the RWP (Read Write Pointer) Drawing Parameter Register.

Table 3 Register Access Commands

Command	Function
ORG	Initialize the relation between the origin point in the X-Y coordinates and the physical address.
WPR	Write into the parameter register
RPR	Read the parameter register
WPTN	Write into the pattern RAM
RPTN	Read the pattern RAM

Table 4 Data Transfer Commands

Command	Function
DRD	DMA read of the frame buffer data
DWT	DMA write into the frame buffer
DMOD	DMA modify of the frame buffer data (bit maskable)
RD	One word read from the frame buffer
WT	One word write into the frame buffer
MOD	One word modify of the frame buffer (bit maskable)
CLR	Clear of frame buffer area
SCLR	Clear of frame buffer area (bit maskable)
CPY	Copy of frame buffer area into another area
SCPY	Copy of frame buffer area into another area (bit maskable)

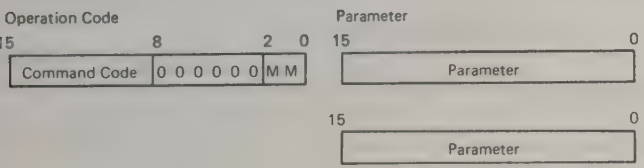


Figure 25 Data Transfer Command Format

MODIFY MODE

The DMOD, MOD, SCLR and SCPY commands allow 4 types of bit level logical operations to be applied to frame buffer data. The modify mode is encoded in the lower two bits (MM) of these op-codes. The bit positions within each frame buffer word to be modified are selectable using the mask register (MASK). Bits masked with 1 are modifiable, those masked with 0 are not.

MM		Modify Mode
0	0	REPLACE frame buffer data with command parameter data.
0	1	OR frame buffer data with command parameter data and rewirte to the frame buffer.
1	0	AND frame buffer data with command parameter data and rewrite to the frame buffer.
1	1	EOR frame buffer data with command parameter data and rewrite to the frame buffer.

• GRAPHIC DRAWING COMMANDS

The ACRTC has 23 separate graphic drawing commands. Graphic drawing is performed by modifying the contents of the frame buffer based upon microcoded drawing algorithms in the ACRTC drawing processor.

Most coordinate parameters for graphic drawing commands are specified using logical pixel X-Y addressing. The complex task of translating a logical pixel address to a linear frame buffer word address, and further selecting the appropriate sub-field of the word (for example, a given logical pixel in 4 bits per logical pixel mode might reside in bits 8-11 of a frame buffer word) is performed at high speed by ACRTC hardware.

Many instructions allow specification of X-Y coordinates with either absolute or relative X-Y coordinates (e.g. ALINE and RLINE). In both cases, twos complement numbers are used to represent positive and negative values.

(a) Absolute Coordinate Specification
The screen address (X, Y) is specified in units of logical pixels relative to an origin point defined with the ORG command.

(b) Relative Coordinate Specification
The screen address (dX,dY) is specified in units of logical pixels relative to the current drawing pointer (CP) position.
A graphic drawing command consists of a 16 bit op-code and optionally 0 to 64k 16 bit parameters.
The 16 bit op-code consists of an 8 bit command code, an AREA Mode specifier (3 bits), a Color Mode specifier (2 bits) and an Operation Mode specifier (3 bits).
The Area Mode allows versatile clipping and hitting detection. A drawing area can be defined, and should drawing operations attempt to enter or leave that area, a number of programmable actions can be taken by the ACRTC.
The Color Mode determines whether the Pattern RAM is used indirectly to select Color Registers or is directly used as the color information.
The Operation Mode defines one of eight logical operations to be performed between the frame buffer read data and the color data in the Pattern RAM to determine the drawing data to be rewritten to the frame buffer.

(i) Absolute Coordinate Specification
Specifies the addresses (x, y) based on the origin point set by the ORG command.

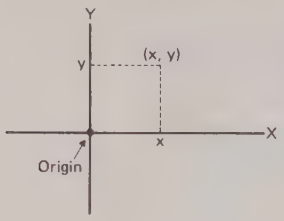


Figure 26 Absolute Coordinate Specification

(ii) Relative Coordinate Specification
Specifies the relative addresses (△x, △y) related to the current drawing point.

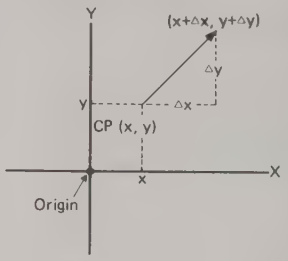


Figure 27 Relative Coordinate Specification

Table 5 Graphic Drawing Commands

Command	Function
AMOVE	Movement of current points
RMOVE	
ALINE	Drawing of straight lines
RLINE	
ARCT	Drawing of rectangles
RRCT	
APLL	Drawing of polylines
RPLL	
APLG	Drawing of polygons
RPLG	
CRCL	Drawing of circles
ELPS	Drawing of ellipses
AARC	Drawing of arcs
RARC	
AEARC	Drawing of ellipse arcs
REARC	
AFRCT	Painting of rectangle areas (Tiling)
RFRCT	
PAINT	Painting of arbitrary areas (Tiling)
DOT	Making of dots
PTN	Drawing of basic patterns (rotation angle: 45°)
AGCPY	Graphic copy between frame memories (rotation angle: 90°/mirror turnover)
RGCPY	

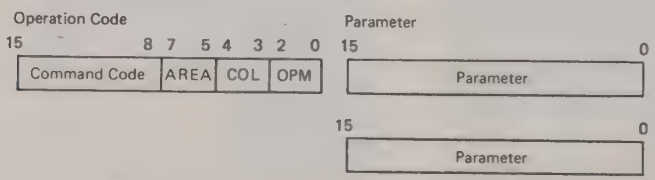


Figure 28 Graphic Drawing Command Format

OPERATION MODE

The Operation Mode (OPM bits) of the Graphic Drawing Command specify the logical drawing condition.

OPM			Operation Mode
0	0	0	REPLACE: Replaces the frame buffer data with the color data.
0	0	1	OR: ORs the frame buffer data with the color data. The result is rewritten to the frame buffer.
0	1	0	AND: ANDs the frame buffer data with the color data. The result is rewritten to the frame buffer.
0	1	1	EOR: EORs the frame buffer data with the color data. The result is rewritten to the frame buffer.
1	0	0	CONDITIONAL REPLACE (P=CCMP): When the frame buffer data at the drawing position (P) is equal to the comparison color (CCMP), the frame buffer data is replaced with the color data.
1	0	1	CONDITIONAL REPLACE (P≠CCMP): When the frame buffer data at the drawing position (P) is not equal to the comparison color (CCMP), the frame buffer data is replaced with the color data.
1	1	0	CONDITIONAL REPLACE (P<CL): When the frame buffer data at the drawing position (P) is less than the color register data (CL), the frame buffer data is replaced with the color data.
1	1	1	CONDITIONAL REPLACE (P>CL): When the frame buffer data at the drawing position (P) is greater than the color register data (CL), the frame buffer data is replaced with the color data.

Figure 29 shows examples of a drawing pattern applied with various OPM modes.

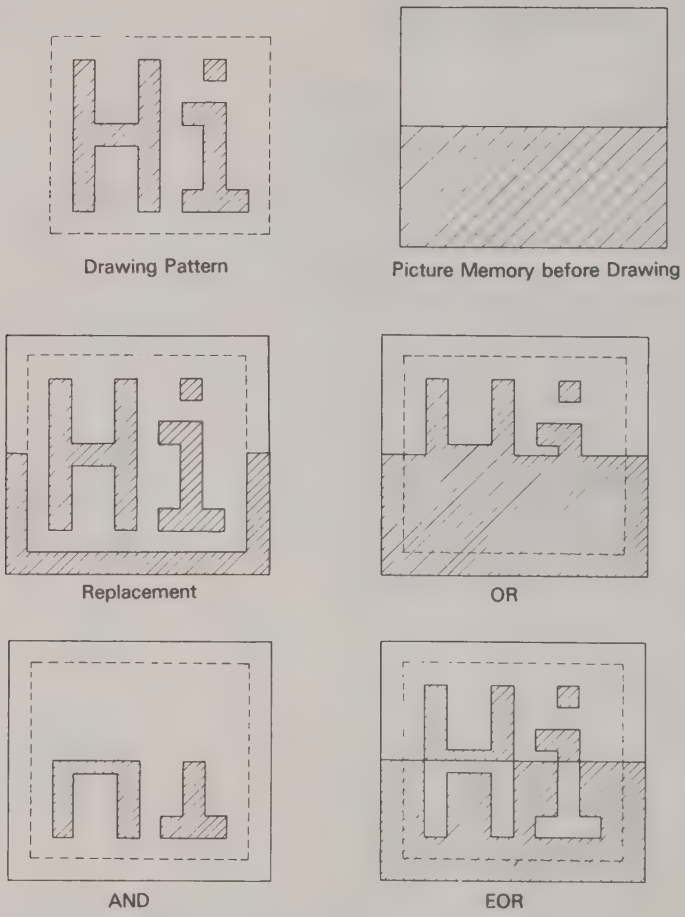


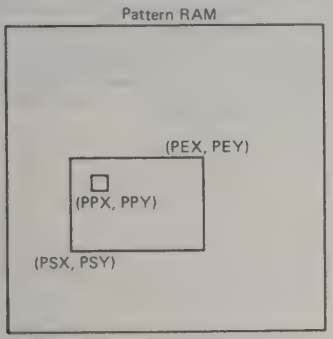
Figure 29 Operation Mode Example

COLOR MODE

The Color Mode (COL bits) specify the source of the drawing color data as directly or indirectly (using the Color Registers) determined by the contents of the Pattern RAM.

COL		Color Mode
0	0	When Pattern RAM data = 0, Color Register 0 is used. When Pattern RAM data = 1, Color Register 1 is used.
0	1	When Pattern RAM data = 0, drawing is suppressed. When Pattern RAM data = 1, Color Register 1 is used.
1	0	When Pattern RAM data = 0, Color Register 0 is used. When Pattern RAM data = 1, drawing is suppressed.
1	1	Pattern RAM contents are directly used as color data.

The Color Mode chooses the source for color information based on the contents (0 or 1) of a particular bit in the 16 bit by 16 bit (32 byte) Pattern RAM. A sub-pattern is specified by programming the Pattern RAM Control Register (PRC) with the start (PSX, PSY) and end (PEX, PEY) points which define the diagonal of the sub-pattern. Furthermore, a specific starting point for Pattern RAM scanning is specified by PPX and PPY.



Normally, the color registers (CL) should be loaded with one color data based on the number of bits per pixel. For example, if 4 bits/pixel are used, the 4 bit color pattern (e.g. 0001) should be replicated four times in the color register, i.e.

Color Register = 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1

In this way, color changes due to changing dot address are avoided.

AREA MODE

Prior to drawing, a drawing 'area' may be defined (Area Definition Register). Then, during Graphics Drawing operation the ACRTC will check if the drawing point is attempting to enter or exit the defined drawing area. Based on eight Area Modes, the ACRTC will take appropriate action for clipping or hitting.

AREA			Drawing Area Mode
0	0	0	Drawing is executed without Area checking.
0	0	1	When attempting to exit the Area, drawing is stopped and the ARD (Area Detect) and CED (Command End) flags are set.
0	1	0	Drawing suppressed outside the Area – drawing operation continues and the ARD flag is not set.
0	1	1	Drawing suppressed outside the Area – drawing operation continues and the ARD flag is set
1	0	0	Same as AREA = 0 0 0.
1	0	1	When attempting to enter the Area, drawing is stopped and the ARD and CED (Command End) flags are set
1	1	0	Drawing suppressed inside the Area – drawing operation continues and the ARD flag is not set
1	1	1	Drawing suppressed inside the Area – drawing operation continues and the ARD flag is set

■ SYSTEM INTERFACE

• BASIC CLOCK

The ACRTC basic clock is 2CLK. 2CLK controls all primary ACRTC display and logic timing parameters.

2CLK, along with the specification of number of bits per logical pixel, the Graphic Address Increment mode and the Display Access mode, also determines the video data rate.

• CRT INTERFACE

FRAME BUFFER ACCESS

(1) Access Modes

The three ACRTC display memory access modes are Single, Interleaved and Superimposed.

(a) Single Access Mode

A display (or drawing) cycle is defined as two cycles of 2CLK. During the first 2CLK cycle, the frame buffer display or drawing address is output. During the second 2CLK cycle, the frame buffer data is read (display cycles and/or drawing cycles) or written (drawing cycles).

In this mode, display and drawing cycles contend for access to the frame buffer. The ACRTC allows the priority to be defined as display priority or drawing priority. If display priority, drawing cycles are only allowed to occur during vertical retrace. So, a 'flashless' display is obtained at the expense of slower drawing. If drawing priority, drawing may occur during display so high speed drawing is obtained, however the display may flash.

(b) Interleaved Access Mode (Dual Access Mode 0)

In this mode, display cycles and drawing cycles are interleaved. A display/drawing cycle is defined as four cycles of 2CLK. During the first 2CLK cycle, the frame buffer display address is output. During the second 2CLK cycle, the display data is read from the frame buffer. During the third 2CLK cycle, the frame buffer drawing address is output. During the fourth 2CLK cycle, the drawing data is read or written.

Since there is no contention between display and drawing cycles, a 'flashless' display is obtained while maintaining full drawing speed. However, for a given configuration, frame buffer memory access time must be twice as fast as an equivalent Single Access Mode configuration.

(c) Superimposed Access Mode (Dual Access Mode 1)

In this mode, two separate logical screens are accessed during each display cycle. The display cycle is defined as four 2CLK cycles. During the first 2CLK cycle, the Background (Upper, Base or Lower) screen frame buffer address is output. During the second 2CLK cycle, the Background screen display or drawing data is read (display or drawing) or written (drawing). During the third 2CLK cycle, the window screen frame buffer address is output. During the fourth 2CLK cycle, the window screen display or drawing data is read (display or drawing) or written (drawing). Note that the third and fourth cycles can be used for Background screen drawing (similar to Interleaved mode) when these cycles are not used for Window display.

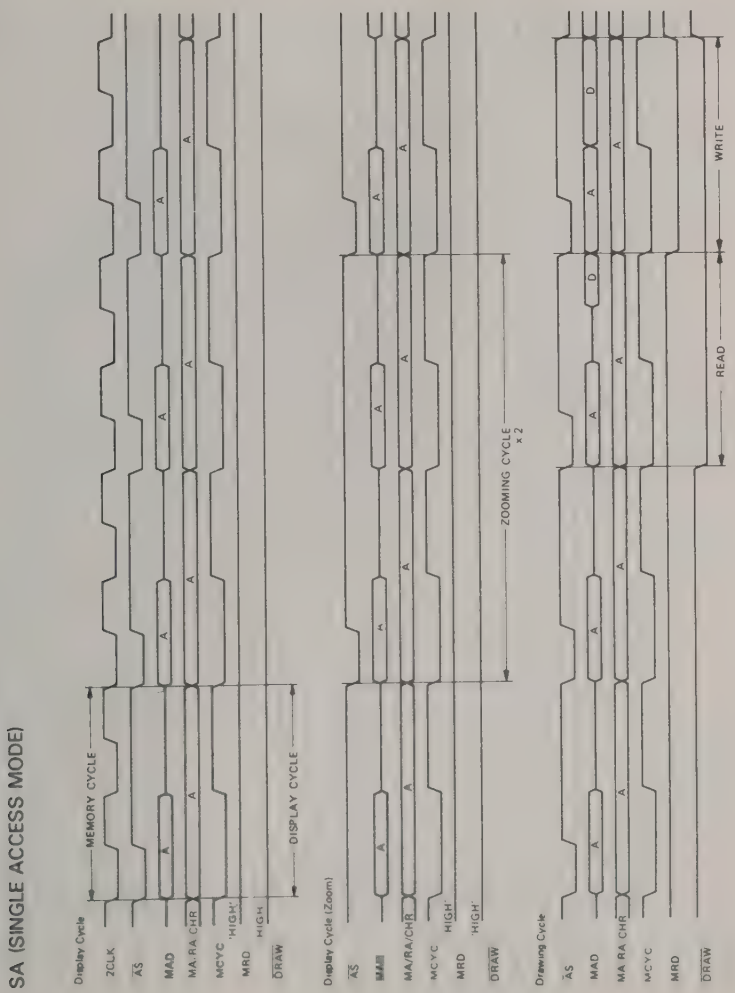


Figure 30(a) Access Mode Timing

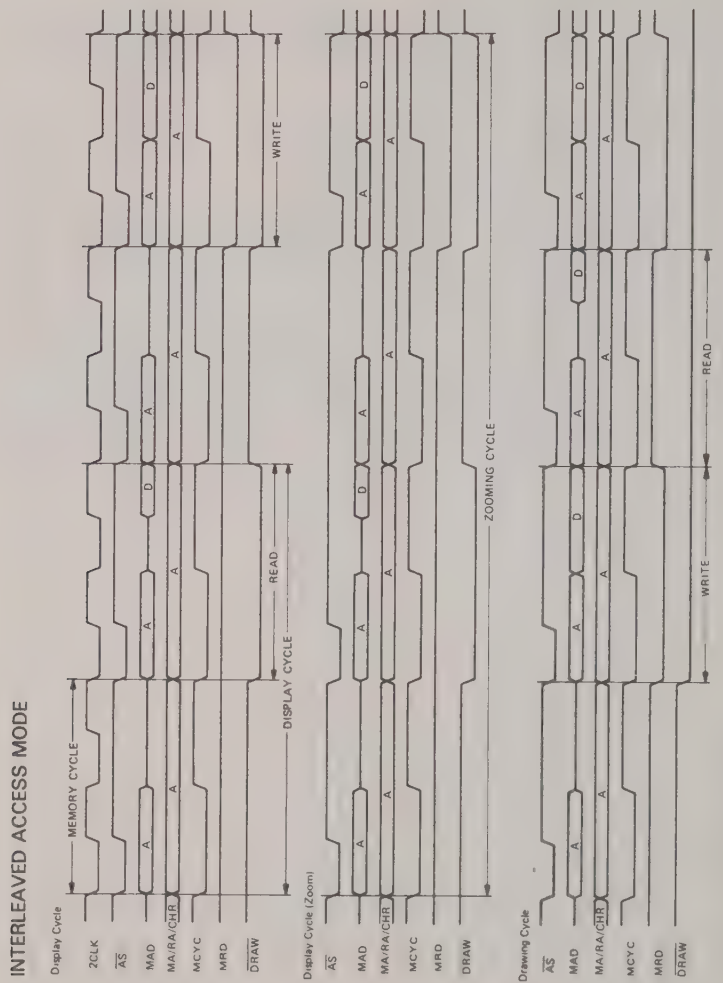


Figure 30(b) Access Mode Timing

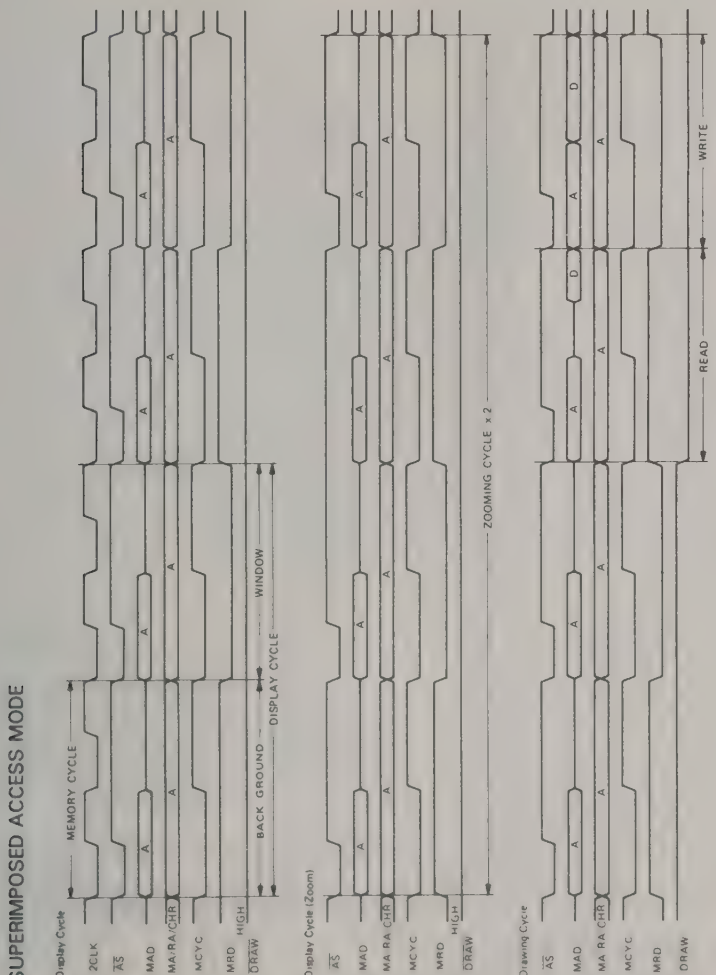


Figure 30(c) Access Mode Timing

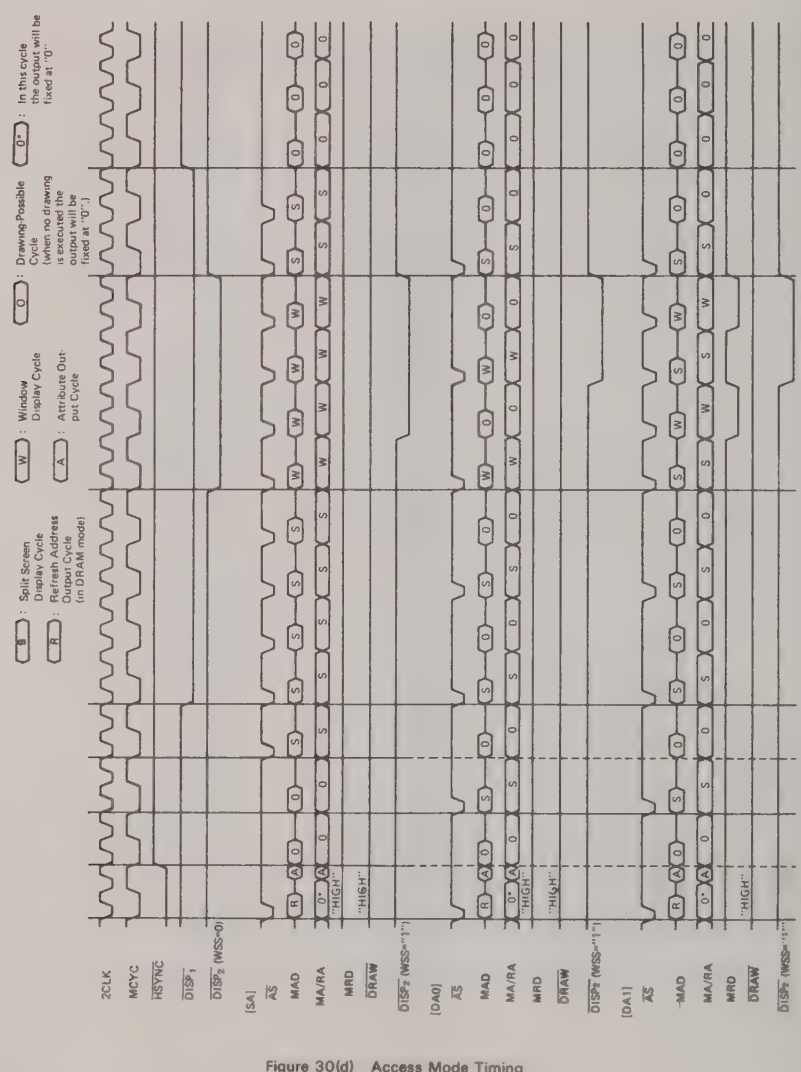


Figure 30(d) Access Mode Timing

(2) Graphic Address Increment Mode (GAI)

During display operation, the ACRTC can be programmed to control the graphic display address in seven ways including increment by 1, 2, 4, 8 and 16* words, 1 word every two display cycles and no increment.

Setting GAI to increment by 2, 4, 8 or 16* words per display cycle achieves linear increases in the video data rate i.e. for a given configuration setting GAI to 2, 4, 8 or 16* words will achieve 2, 4, 8 or 16* times the video data rate corresponding to GAI=1. This allows increasing the number of bits/logical pixel and logical pixel resolution while meeting the 2CLK maximum frequency constraint.

Table 6 shows the summary relationship between 2CLK, Display Access Mode, Graphic Address Increment, # bits/logical pixel, memory access time and video data rate. The frame buffer cycle frequency (Fc) is shown by the following equation where:

Fv = Dot Clock
N = # bits/logical pixel
D = Display Access Mode
1 for Single Access Mode
2 for interleaved and Superimposed Access Modes
A = Graphic Address Increment (1/2, 1, 2, 4, 8, 16*)
Fc = (Fv × N × D)/(A × 16)

Table 6 Graphic Address Increment Modes

Dot Rate		16MHz		32MHz		64MHz		128MHz	
Access Mode Color No. (bit/pixel)	Memory Cycle	S	D	S	D	S	D	S	D
1	250ns	–	+1/2	+1/2	+1	+1	+2	+2	+4
	500ns	+1/2	+1	+1	+2	+2	+4	+4	+8
2	250ns	+1/2	+1	+1	+2	+2	+4	+4	+8
	500ns	+1	+2	+2	+4	+4	+8	+8	+16*
4	250ns	+1	+2	+2	+4	+4	+8	+8	+16*
	500ns	+2	+4	+4	+8	+8	+16*	+16*	–
8	250ns	+2	+4	+4	+8	+8	+16*	+16*	–
	500ns	+4	+8	+8	+16*	+16*	–	–	–
16	250ns	+4	+8	+8	+16*	+16*	–	–	–
	500ns	+8	+16*	+16*	–	–	–	–	–

DYNAMIC RAM REFRESH

When dy ic RAMs (DRAMs) are used for the frame buffer mem the ACRTC can automatically provide DRAM refresh addressing.

The ACRTC maintains an 8 bit DRAM refresh counter which is decremented on each frame buffer access. During HSYNC low, the ACRTC will output the sequential refresh addresses on MAD. The refresh address assignment depends on Graphic Address Increment (GAI) mode as shown in Table 7.

The ACRTC provides "0" output on the remaining address line of MAD and MA/RA.

DRAM refresh cycle timing must be factored into the determination of HSYNC low pulse width (HSW – specified in units of frame buffer memory cycles).

If the horizontal scan rate is Fh (kHz), number of DRAM refresh cycles is N and the DRAM refresh cycle time is Tr (msec) then horizontal sync width (HSW) is specified by the following equation:

Table 7 GAI and DRAM Refresh Addressing

Address Increment Mode	Refresh Address Output Terminal
+0 (GAI=101)	MAD_0~MAD_7
+1 (GAI=000)	MAD_0~MAD_7
+2 (GAI=001)	MAD_1~MAD_8
+4 (GAI=010)	MAD_2~MAD_9
+8 (GAI=011)	MAD_3~MAD_{10}
+16 (GAI=100)*	MAD_4~MAD_{11}
+1/2 (GAI=$^{111}_{110}$)	MAD_0~MAD_7

$HSW \geq N / (Tr \times Fh)$

For example, if the scan rate is 15.75 kHz and the DRAMS have 128 refresh cycles of 2 msec, HSW must be greater than or equal to 5.

$HSW \geq 128 / (2 \times 15.75) = 4.06$

(Note) * ... HD63484 (R-type) does not support 16 words increment mode

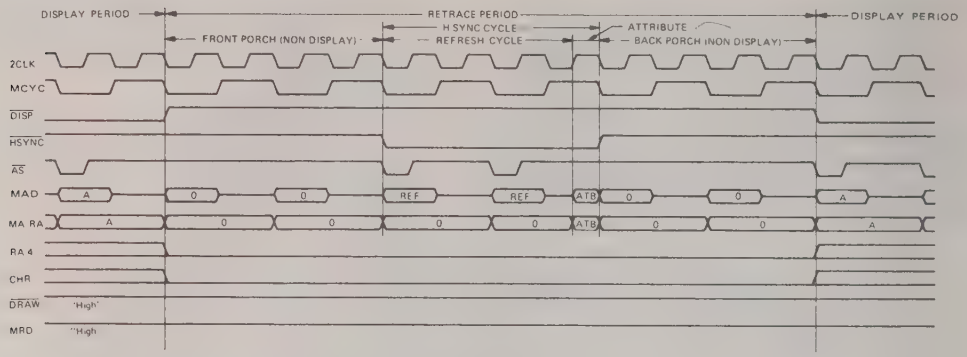


Figure 31 DRAM Refresh Timing

EXTERNAL SYNCHRONIZATION

The ACRTC $\overline{\text{EXSYNC}}$ pin allows synchronization of multiple ACRTCs or other video signal generators. The ACRTC may be programmed as a single Master device, or as one of a number of Slave devices.

To synchronize multiple ACRTCs, simply connect all the $\overline{\text{EXSYNC}}$ pins together.

For synchronizing to other video signals, the connection scheme depends on the raster scan mode. In Non-Interlace mode, $\overline{\text{EXSYNC}}$ corresponds to $\overline{\text{VSYNC}}$. In Interlace modes, $\overline{\text{EXSYNC}}$ corresponds to $\overline{\text{VSYNC}}$ of the odd field.

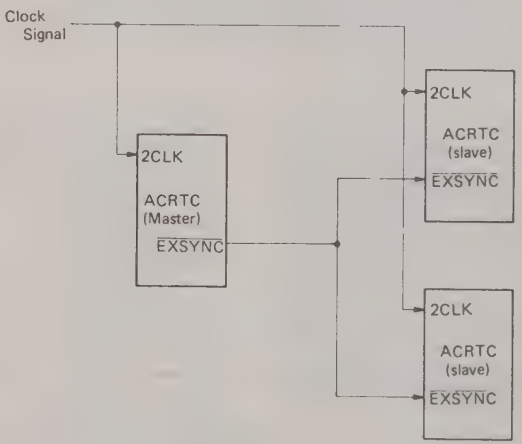


Figure 32 External Synchronization

• MPU INTERFACE

MPU BUS CYCLE

The ACRTC interfaces to the MPU as a peripheral occupying two addresses in the MPU address space. The ACRTC can operate as an 8 or 16 bit peripheral as configured during $\overline{\text{RES}}$.

An MPU bus cycle is initiated when $\overline{\text{CS}}$ is asserted (following the assertion of RS and R/$\overline{\text{W}}$). The ACRTC responds to $\overline{\text{CS}}$ low by asserting $\overline{\text{DTACK}}$ low to complete the data transfer. $\overline{\text{DTACK}}$ will be returned to the MPU in between 1 and 1.5 2CLK cycles.

MPU WAIT states will be added in the following two cases.

(a) If the ACRTC 2CLK input is much slower than the MPU clock, continuous ACRTC accesses may be delayed due to internal processing of the previous bus cycle.

(b) If an ACRTC read cycle immediately follows an ACRTC write cycle, a WAIT state may occur due to ACRTC preparation for bus 'turn-around'. However, MPUs normally have no instructions which immediately follow a write cycle with a read cycle.

For connection to synchronous bus interface MPUs, $\overline{\text{DTACK}}$ can simply be left open assuming the system design guarantees

that WAIT states cannot occur as described above. If WAIT states may occur, $\overline{\text{DTACK}}$ can be used with external logic to synthesize a READY signal.

DMA TRANSFER

The ACRTC can interface with an external DMA controller using three handshake signals, DMA Request ($\overline{\text{DREQ}}$), DMA Acknowledge ($\overline{\text{DACK}}$) and DMA Done ($\overline{\text{DONE}}$).

The ACRTC uses the external DMAC for two types of transfers, Command/Parameter DMA and Data DMA. For both types, DMA transfers use the ACRTC read and write FIFOs.

(1) Command/Parameter DMA

The MPU initiates this mode by setting bit 12 (CDM: Command DMA Mode) in the ACRTC Command Control Register (CCR) to 1. Then, the ACRTC will automatically request DMA transfer for commands and their associated parameters as long the write FIFO has space. Only cycle steal request mode ($\overline{\text{DREQ}}$ pulses low for each data transfer) can be used. Command/Parameter DMA is terminated when the MPU resets bit 12 in CCR to 0 or the external $\overline{\text{DONE}}$ input is asserted.

(2) Data DMA

Data DMA is used to move data between the MPU system memory and the ACRTC frame buffer.

The MPU sets-up the transfer by specifying the frame buffer transfer address (and other parameters of the transfer, such as 'on-the fly' logical operations) to the ACRTC. Next, when the MPU issues a Data Transfer Command to the ACRTC, the ACRTC will request DMA transfer to and from system memory. The ACRTC will request DMA, automatically monitoring FIFO status, until the DMA Transfer Command is completed.

Data DMA request mode can be cycle steal (as in Command/Parameter DMA) or burst mode in which $\overline{\text{DREQ}}$ is a low level control output to the DMAC which allows multiple data transfers during each acquisition of the MPU bus.

INTERRUPTS

The ACRTC recognizes eight separate conditions which can generate an interrupt including command error detection, command end, drawing edge detection, light pen strobe and four FIFO status conditions. Each condition has an associated mask bit for enabling/disabling the associated interrupt. The ACRTC removes the interrupt request when the MPU performs appropriate interrupt service by reading or writing to the ACRTC.

■ DISPLAY FUNCTION

• LOGICAL DISPLAY SCREENS

The ACRTC allows division of the frame buffer into four separate logical screens.

Screen Number	Screen Name	Screen Group Name
0	Upper Screen	Background Screens
1	Base Screen	
2	Lower Screen	
3	Window Screen	

In the simplest case, only the Base screen parameters must be defined. Other screens may be selectively enabled, disabled and blanked under software control.

The Background (Upper, Base and Lower) screens partition the display into three horizontal splits whose position is fully programmable. A typical application might use the Base screen for the bulk of user interaction, using the Lower screen for a 'status line(s)' and the Upper screen for 'pull-down menu(s)'.

The Window screen is unique, since the ACRTC gives the Window screen higher priority than Background screens. thus, when the Window, whose size and position is fully programmable, overlaps a Background screen, the Window screen is displayed. One exception is the ACRTC Superimposed Access Mode, in which the Window has the same display priority as Background screens. In this case, the Window and Background screen are 'superimposed' on the display.

The ACRTC logical screen organization can be programmed to best suit a number of display applications.

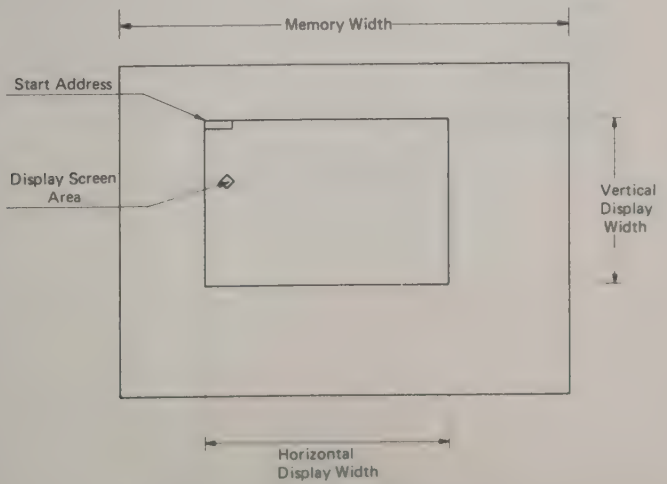


Figure 33 Display Screen/Frame Buffer Relationship

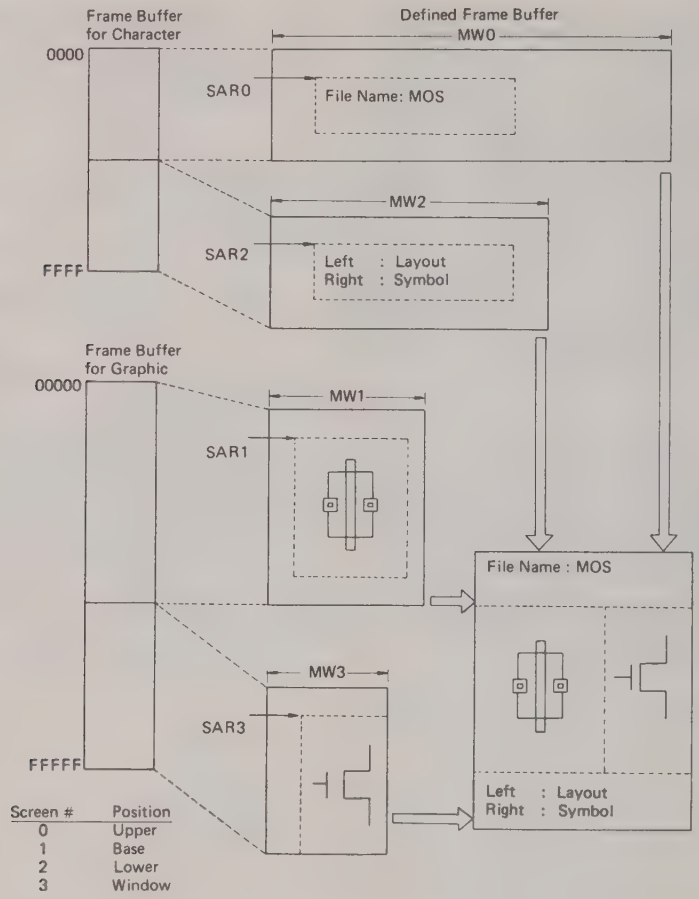


Figure 34 Display Screen Combination

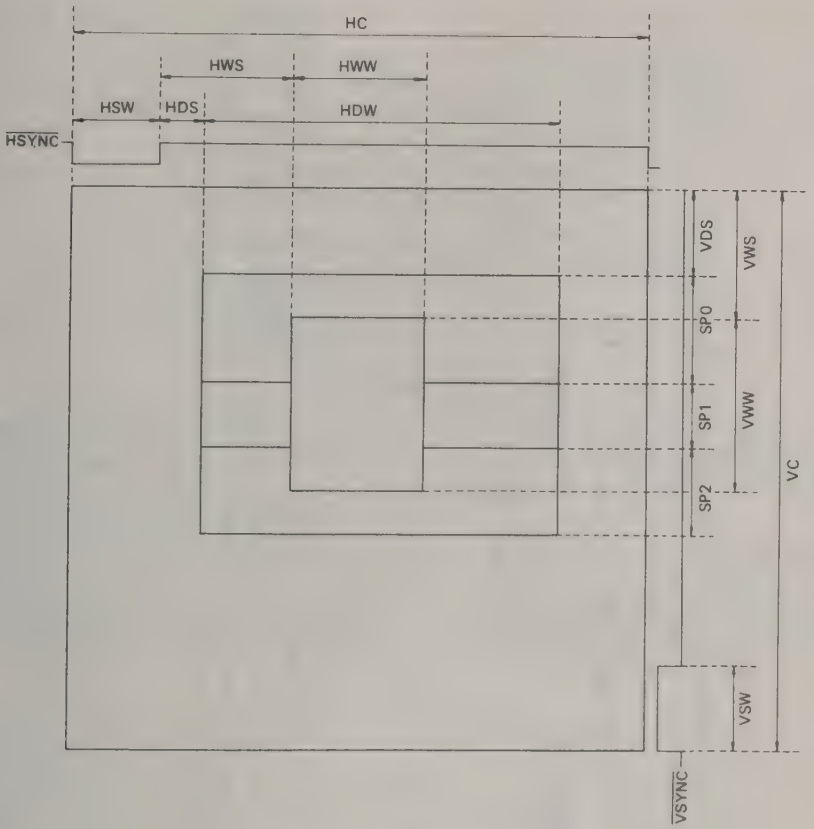


Figure 35 Display Screen Specification

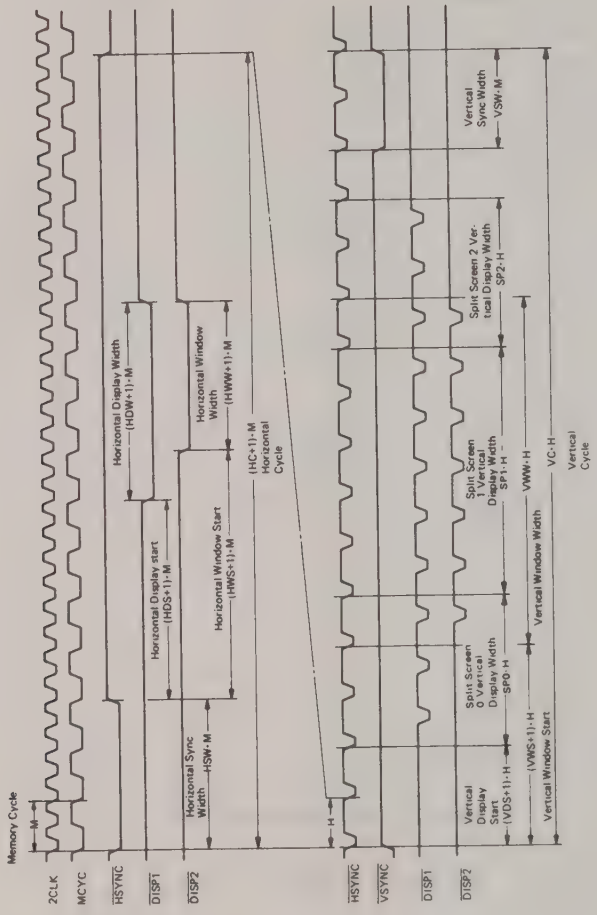


Figure 36 Display Screen Timing

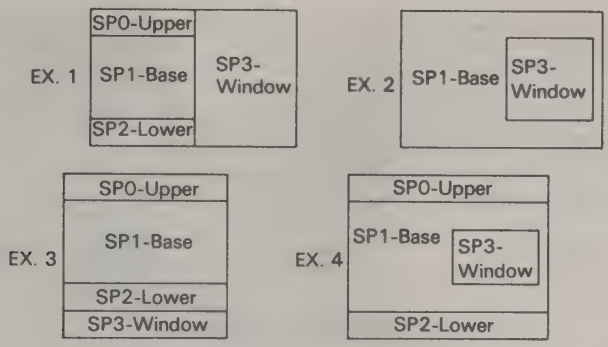


Figure 37 Example Screen Combinations

GRAPHIC/CHARACTER ADDRESS SPACES

The ACRTC controls two separate logical address spaces. The CHR pin allows external decoding if physically separate frame buffers are desired.

Each of the four logical screens (Upper, Base, Lower and Window) is programmed as residing in the Graphics address space or the Character address space.

ACRTC accesses to Graphics screens are treated as bit mapped using a 20 bit frame buffer address, with an address space of one megaword (1M by 16 bit).

ACRTC accesses to Character screens are treated as character generator mapped. In this case, a 64k word address space is used and 5 bits of raster address are output to an external character generator.

Multiple logical screens defined as Character can be externally decoded to use separate character generators or different addresses within a combined character generator. Also, each Character screen may be defined with separate line spacing, separate cursors, etc.

- **CURSOR CONTROL**

The ACRTC has two Block Cursor Registers and a Graphics Cursor Register.

A Block cursor is used with Character screens. The cursor start and ending raster addresses are fully programmable. Also, the cursor width can be defined as one to eight memory cycles.

A Graphics cursor is defined by specifying the start and end addresses in both the X and Y dimensions.

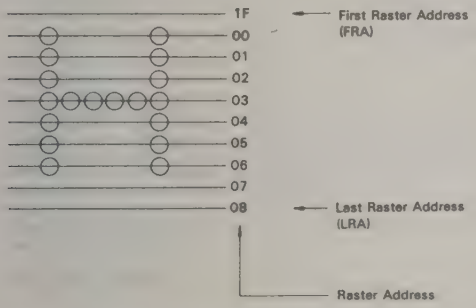


Figure 38 Character Screen Raster Addressing

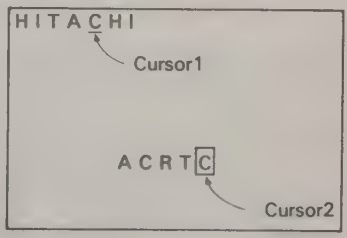


Figure 39 Two Separate Block Cursors

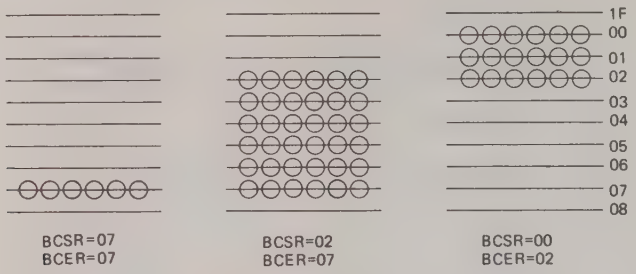


Figure 40 Block Cursor Examples

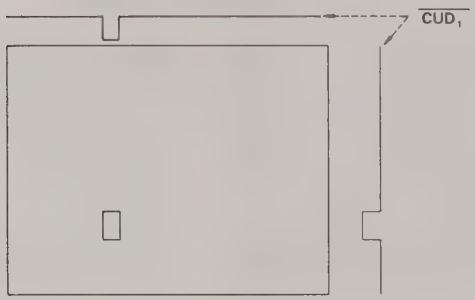


Figure 41 Graphic Cursor

The ACRTC provides two separate cursor outputs, $\overline{CUD_1}$ and $\overline{CUD_2}$. These are combined with two character cursor registers and a graphics cursor register to provide three cursor modes.

Block Mode

Two Block cursors are output on $\overline{CUD_1}$ and $\overline{CUD_2}$ respectively.

Graphic Mode

The Graphic cursor is output on $\overline{CUD_1}$. Using an external cursor pattern memory allows a graphic cursor of various shapes. Two Block cursors are multiplexed on $\overline{CUD_2}$.

Crosshair Mode

The horizontal and vertical components of the Graphic cursor are output on $\overline{CUD_1}$ and $\overline{CUD_2}$ respectively. This allows simple generation of a crosshair cursor control signal.

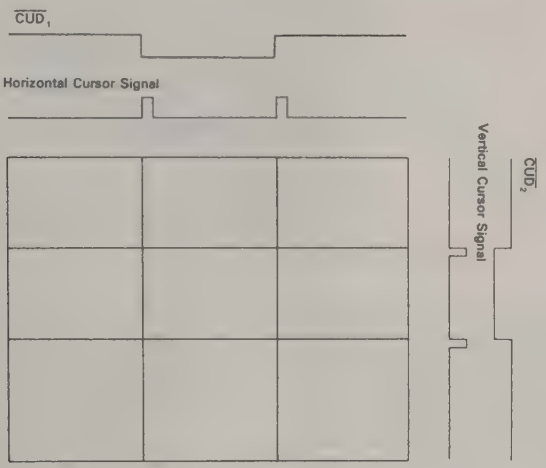


Figure 42 Crosshair Cursor

• SCROLLING

VERTICAL SCROLL

Each logical screen performs independent vertical scroll. On Character Screens, vertical smooth scroll is accomplished using the programmable Start Raster Address (SRA). Line by line scroll is accomplished by increasing or decreasing the screen start address by one unit of horizontal memory width.

On Graphics screens, vertical smooth scroll is accomplished by increasing or decreasing the screen start address by one unit of horizontal memory width.

HORIZONTAL SCROLL

Horizontal scroll can be performed in units of characters for Character screens and units of words (multi logical pixels) for Graphic screens by increasing or decreasing the screen start address by 1.

For smooth horizontal scroll, the ACRTC has dot shift video attributes which can be used with an external circuit which conditions shift register load/clocking.

Since this dot shift information is output each raster, horizontal smooth scroll is limited to either the Background screens or the Window screen at any given time. However, horizontal smooth scroll is independent for each of the Background screens (Upper, Base, Lower).

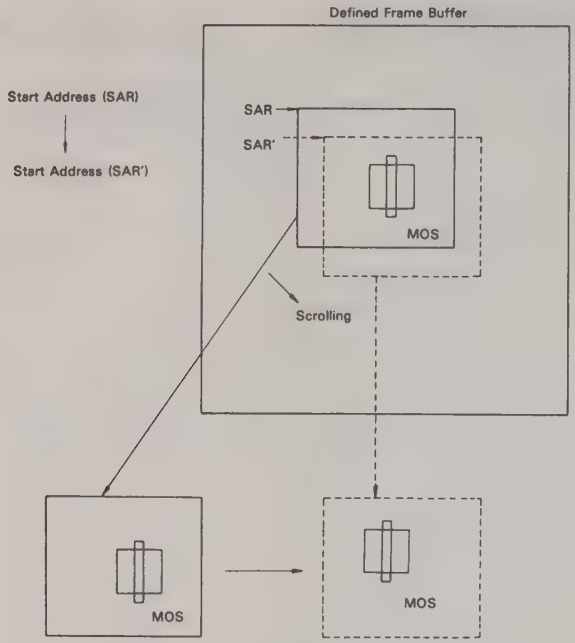


Figure 43 Scrolling By SAR (Start Address Register) Rewrite

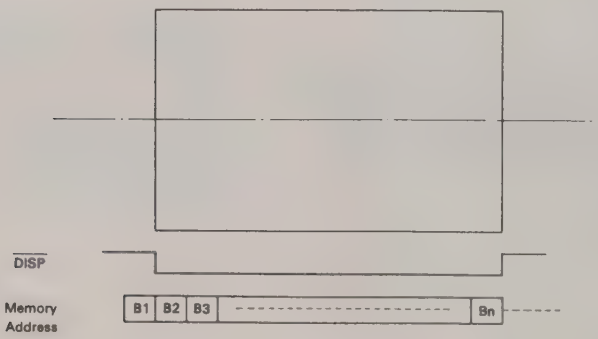


Figure 44 Horizontal Smooth Scroll — Base Screen

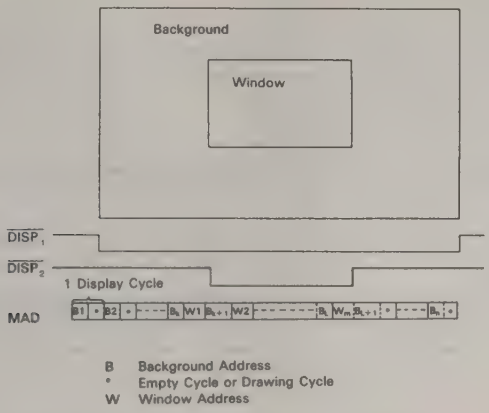


Figure 45 Horizontal Smooth Scroll — Window Screen

• RASTER SCAN MODES

The ACRTC has three software selectable raster scan modes — Non-Interlace, Interlace Sync and Interlace Sync & Video. In Non-Interlace mode a frame consists of one field. In the Interlace modes, a frame consists of two fields, the even and odd fields.

The Interlace modes allow increasing screen resolution while avoiding limits imposed by the CRT display device, such as maximum horizontal scan frequency or maximum video dot rate.

Interlace Sync mode simply repeats each raster address for both the even and odd fields. This is useful for increasing the quality of a displayed figure when using an interlaced CRT device such as a Television Set with RF modulator.

Interlace Sync & Video mode displays alternate even and odd rasters on alternate even and odd fields. For a given number of rasters/character, this mode allows twice as many characters to be displayed in the vertical direction as Non-Interlace mode.

Note that for Interlace modes, the refresh frequency for a given dot on the screen is one-half that of the Non-Interlace mode. Interlace modes normally require the use of a CRT with a more persistent phosphor to avoid a flickering display.

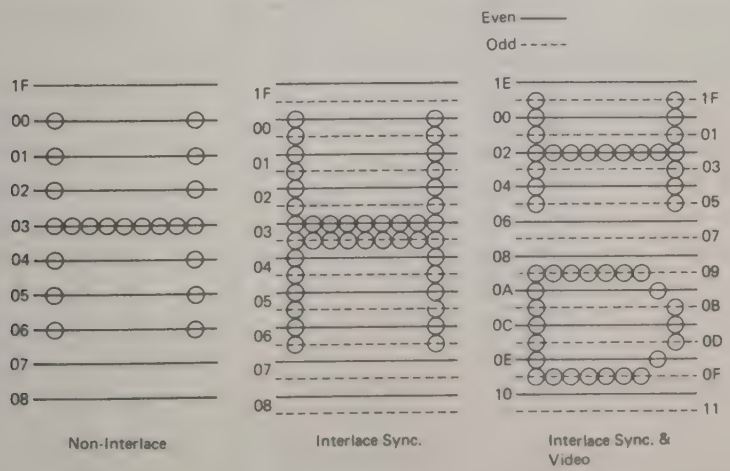


Figure 46 Raster Scan Modes

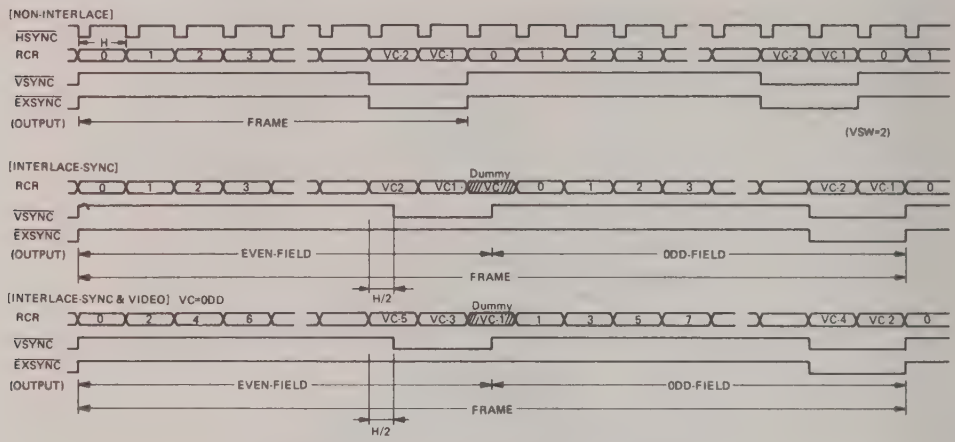


Figure 47 Raster Scan Timing

• ZOOMING

The Base screen (Screen 1) is supported by the ACRTC zooming function. Note that ACRTC zooming is performed by controlling the CRT timing signals. The contents of the frame buffer area being zoomed are not changed.

The ACRTC allows specification of a zoom factor (1 to 16) independently in the X and Y directions.

For horizontal zoom, the programmed zoom factor is output as video attributes. An external circuit uses this factor to condition the external shift register clock to accomplish horizontal zooming.

For vertical zoom, no external circuit is required. The ACRTC will scan a single raster multiple times to accomplish vertical zooming.

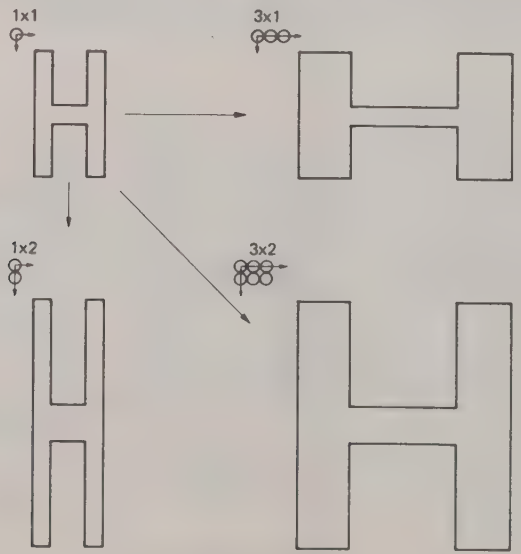


Figure 48 Zooming

- **LIGHT PEN**

The ACRTC provides a 20 bit Light Pen Address Register and a Light Pen Strobe (LPSTB) input pin for connection with a light pen.

A light pen strobe pulse will occur when the CRT electron beam passes under the light pen during display refresh. When this pulse occurs, the contents of the ACRTC display refresh address counter will be latched into the Light Pen Address Register along with a logical screen (Character or Graphic screen) designator. Also, an ACRTC status flag indicating light pen activity is set, generating an optional (maskable) MPU interrupt. Note that for Superimposed access mode, when the light pen strobe occurs in an area in which the Window overlaps a Background (Upper, Base or Lower) screen, the Background screen address will be latched.

Various system and ACRTC delays will cause the latched address to differ slightly from the actual light pen position. the light pen address can be corrected using software, based upon system specific delays. Or, if the application does not require the highest light pen pointing resolution, software can 'bound' the light pen address by specifying a range of values associated with a given area of the screen.

INTRODUCTION OF THE RELATED DEVICES

- 4-bit Single-chip Microcomputer
- 8-bit Single-chip Microcomputer
- IC Memory
- LCD Driver Series
- Gate Array
- CODEC/Filter Combo LSI
- Speech Synthesis LSI

4-BIT SINGLE-CHIP MICROCOMPUTER

■ PMOS 4-BIT SINGLE-CHIP MICROCOMPUTER HMCS40 SERIES

	Family Name (Type Name)		HMCS42 (HD38702)	HMCS43 (HD38750, HD38755)	HMCS44A (HD38800, HD38805)	HMCS45A (HD38820, HD38825)	HMCS47A (HD38870)
LSI Characteristics	Supply Voltage (V)		-10	-10	-10	-10	-10
	Power Dissipation (typ.) (mW)		100	100	150	150	250
	Max. I/O Terminal Voltage (V)		-50	-50	-50	-50	-50
	Operating Temperature Range *1 (°C)		-20 to +75	-20 to +75	-20 to +75	-20 to +75	-20 to +75
	Package		DP-28, DP-28S	DP-42, DP-42S	DP-42, DP-42S	FP-54, DP-64S	FP-54, DP-64S
Functions	Memory	ROM (bits)	512 x 10 32 x 10 *2	1,024 x 10 64 x 10 *2	2,048 x 10 128 x 10 *2	2,048 x 10 128 x 10 *2	4,096 x 10
		RAM (bits)	32 x 4	80 x 4	160 x 4	160 x 4	256 x 4
	Registers		4	6	8	6	6
	Stack Registers		2	3	4	4	4
	I/O Ports	4-Bit Data Input	22	32	32	44	44
		4-Bit Data Output	4 x 1	4 x 1	–	–	–
		Discrete Output	4 x 2	4 x 2	–	4 x 1	4 x 1
		4-Bit Data Input/Output	1 x 6	1 x 12	–	–	–
		Discrete Input/Output	–	4 x 1	4 x 4	4 x 6	4 x 6
			1 x 4	1 x 4	1 x 16	1 x 16	1 x 16
	Interrupts	External	–	2	2	2	2
		Timer/Counter	–	1	1	1	1
	Instructions	Number of Instructions	51	71	71	71	71
		Cycle Time (μs)	10	10	10	10	10
	Built-in Clock Pulse Generator		Yes				
	Power on Reset		Yes	Yes	Yes	Yes	Yes
	Battery Back-up		–	RAM Hold	RAM Hold	RAM Hold	RAM Hold
Evaluation Chip			HD38750E HD44850E HD44857E	HD38750E HD44850E HD44857E	HD44850E HD44857E	HD44850E HD44857E	HD44857E

*1 Wide Temperature Range (-40 to +85°C) version is available.
*2 Pattern Memory

■ CMOS 4-BIT SINGLE-CHIP MICROCOMPUTER HMCS40 SERIES

	Family Name (Type Name)			HMCS42CL (HD44708) HMCS42C (HD44700)		HMCS43CL (HD44758) HMCS43C (HD44750)		HMCS44CL (HD44808) HMCS44C (HD44801)	
LSI Characteristics	Supply Voltage		(V)	3/5		3/5		3/5	
	Power Dissipation (typ.)		(mW)	0.23/1.5		0.24/1.5		0.32/2	
	Max. I/O Terminal Voltage		(V)	V_{CC}+0.3		V_{CC}+0.3		V_{CC}+0.3	
	Operating Temperature Range *1		(°C)	-20 to +75		-20 to +75		-20 to +75	
	Package			DP-28, DP-28S		DP-42, DP-42S		DP-42, DP-42S	
Functions	Memory	ROM	(bits)	512 x 10 32 x 10*2		1,024 x 10 64x10*2		2,048 x 10 128x10*2	
		RAM	(bits)	32 x 4		80 x 4		160 x 4	
	Registers			4		6		8	
	Stack Registers			2		3		4	
	I/O Ports	4-Bit Data Input		22	4 x 1	32	4 x 1	32	–
		4-Bit Data Output			4 x 2		4 x 2		–
		Discrete Output			1 x 6		1 x 12		–
		4-Bit Data Input/Output			–		4 x 1		4 x 4
		Discrete Input/Output			1 x 4		1 x 4		1 x 16
	Interrupts	External		–		2		2	
		Timer/Counter		–		1		1	
	Instructions	Number of Instructions		51		71		71	
		Cycle Time	(μs)	20/10		20/10		20/10	
	Built-in Clock Pulse Generator			Yes					
	Power on Reset			No/Yes		No/Yes		No/Yes	
	Battery Back-up			Halt		RAM Hold		Halt	
Evaluation Chip				HD44850E HD44857E		HD44850E HD44857E		HD44850E HD44857E	

*1 Wide Temperature Range (-40 to +85°C) version is available.
*2 Pattern Memory
*3 LCD DRIVE FUNCTION

LCD Drive	Common	4
	Segment	32
	Duty	Static, 1/2, 1/3, 1/4
	Bias	1/2, 1/3
Display Capability		4x32 Matrix (1/4 Duty)

Expandable using the LCD Driver HD44100H.

HMCS45CL (HD44828) HMCS45C (HD44820)	HMCS46CL (HD44848) HMCS46C (HD44840)	HMCS47CL (HD44868) HMCS47C (HD44860)	LCD-III*3 (HD44795, HD44790)	LCD-IV*3 (HD613901)
3/5	3/5	3/5	3/5	3/5
0.32/2	0.32/4	0.32/4	0.36/2.4	0.9/5.0
V_{CC}+0.3	V_{CC}+0.3	V_{CC}+0.3	V_{CC}+0.3	V_{CC}+0.3
-20 to +75	-20 to +75	-20 to +75	-20 to +75	-20 to +75
FP-54, DP-64S	DP-42, DP-42S	FP-54, DP-64S	FP-80	FP-80
2,048 x 10 128x10*3	4,096 x 10	4,096 x 10	2,048 x 10 128 x 10*3	4,096 x 10
160 x 4	256 x 4	256 x 4	160 x 4	256 x 4
6	8	6	6	6
4	4	4	4	4
44 –	32 –	44 –	32 4 x 1	32 4 x 1
4 x 1	–	4 x 1	4 x 1	4 x 1
–	–	–	–	–
4 x 6	4 x 4	4 x 6	4 x 2	4 x 2
1 x 16	1 x 16	1 x 16	1 x 16	1 x 16
2	2	2	2	2
1	1	1	1	1
71	71	71	71	71
20/10	20/5	20/5	20/10	20/5
Yes				
No/Yes	No/Yes	No/Yes	Yes	No
Halt	Halt	Halt	Halt	Halt
HD44850E HD44857E	HD44857E	HD44857E	HD44797E	HD44797E

■ CMOS 4-BIT SINGLE-CHIP MICROCOMPUTER HMCS400 SERIES

Family Name (Type Name)			HMCS404C (HD614042)	HMCS404AC* (HD614048)	HMCS404CL (HD614045)	HMCS412C** (HD614120)
LSI Characteristics	Supply Voltage	(V)	4 to 6	4.5 to 6.0	2.7 to 6.0	4 to 6
	Power Dissipation	(max.) (mW)	18	27	9	12
	Max. I/O Terminal Voltage	(V)	V_{CC}-40	V_{CC}-40	V_{CC}-40	V_{CC}-40
	Operating Temperature Range	(°C)	-20 to +75	-20 to +75	-20 to +75	-20 to +75
	Package		FP-64, DP-64S	FP-64, DP-64S	FP-64, DP-64S	DP-42
Functions	Memory	ROM (bits)	4096 x 10	4096 x 10	4096 x 10	2048 x 10
		RAM (bits)	256 x 4	256 x 4	256 x 4	160 x 4
	Registers		7	7	7	7
	Stack Registers		16	16	16	16
	I/O Ports	4-Bit Input	58: 4 x 1, 2 x 1	58: 4 x 1, 2 x 1	58: 4 x 1, 2 x 1	36: 1 x 1
		4-Bit Output	58: 4 x 4	58: 4 x 4	58: 4 x 4	36: 4 x 1
		4-Bit Input/Output	58: 4 x 5	58: 4 x 5	58: 4 x 5	36: 4 x 4
		1-Bit Input/Output	58: 1 x 16	58: 1 x 16	58: 1 x 16	36: 1 x 15
	Interrupts	External	2	2	2	2
		Timer/Counter	2	2	2	1
		Serial Interface	1	1	1	--
	Instructions	Number of Instructions	99	99	99	98
		Cycle Time (μs)	2	1.33	4	2
	Built-in Clock Pulse Generator		Yes (External drive is possible)			
	Others		Low Power Dissipation Mode (Stop mode, Stand-by mode)			

Family Name (Type Name)			HD614P080S†	HD614P180**†
LSI Characteristics	Supply Voltage	(V)	4.5 to 5.5	4.5 to 5.5
	Power Dissipation	(max.) (mW)	27	27
	Max. I/O Terminal Voltage	(V)	V_{CC}-40	V_{CC}-40
	Operating Temperature Range	(°C)	-20 to +75	-20 to +75
	Package		DC-64SP	DC-42
Functions	Memory	ROM (bits)	•4,096-word x 10-bit with standard EPROM 2764 •8,192-word x 10-bit with standard EPROM 27128	•4,096-word x 10-bit with standard EPROM 2764 •8,192-word x 10-bit with standard EPROM 27128
		RAM (bits)	576 x 4	576 x 4
	Registers		7	7
	Stack Registers		16	16
	I/O Ports	4-Bit Input	58: 4 x 1, 2 x 1	36: 1 x 1
		4-Bit Output	58: 4 x 4	36: 4 x 1
		4-Bit Input/Output	58: 4 x 5	36: 4 x 4
		1-Bit Input/Output	58: 1 x 16	36: 1 x 15
	Interrupts	External	2	2
		Timer/Counter	2	1
		Serial Interface	1	–
	Instructions	Number of Instructions	99	98
		Cycle Time (μs)	1.33	2
	Built-in Clock Pulse Generator		Yes (External drive is possible)	
	Others		Low Power Dissipation Mode (Stop mode, Stand-by mode)	

* Preliminary
** Under development
† EPROM on the Package Type

8-BIT SINGLE-CHIP MICROCOMPUTER

■ NMOS 8-BIT SINGLE-CHIP MICROCOMPUTER HD6801 SERIES

Type No.			HD6801S0 HD6801S5	HD6801V0 HD6801V5	HD6803 HD6803-1
LSI Characteristics	Bus Timing (MHz)		1.0/1.25	1.0/1.25	1.0/1.25
	Supply Voltage (V)		5.0	5.0	5.0
	Operating Temperature * (°C)		0 ~ +70	0 ~ +70	0 ~ +70
	Package		DP-40	DP-40	DP-40
Functions	Memory	ROM (k byte)	2	4	–
		RAM (byte)	128	128	128
	I/O Port		29	29	13
	Interrupt	External	2	2	2
		Soft	1	1	1
		Timer	3	3	3
		Serial	1	1	1
	Timer		• Free running counter 16-bit x 1 • Output compare register 16-bit x 1 • Input capture register 16-bit x 1		
	SCI		Full double step-stop type		
	External Memory Expansion		• Address/data non-multiple mode (256 bytes) • Address/data multiple mode (65k bytes)		• Address/data multiple mode (65k bytes)
	Clock Pulse Generator		Built-in (External clock useable)		
	Built-in RAM Holding		Yes (64 bytes)		
EPROM on the Package Type**			HD68P01V07 HD68P01V07-1	HD68P01V07 HD68P01V07-1	¬
Compatibility			MC6801 MC6801-1	–	MC6803 MC6803-1

* Wide Temperature Range (-40 ~ +85°C) version is available.

** HD68P01M0 and HD68P01M0-1 are useable.

■ NMOS 8-BIT SINGLE-CHIP MICROCOMPUTER HD6805 SERIES

Type No.			HD6805S1	HD6805S6	HD6805U1
LSI Characteristics	Clock Frequency (MHz)		1.0	1.0	1.0
	Supply Voltage (V)		5.25	5.25	5.25
	Operating Temperature** (°C)		0 ~ +70	0 ~ +70	0 ~ +70
	Package		DP-28	DP-28	DP-40
Functions	Memory	ROM (k byte)	1.1	1.8	2
		RAM (byte)	64	64	96
	I/O Port	I/O Port	20	20	32
		Input Port	20	–	24
					8
	Interrupt	Nesting	6	6	6
		External	1	1	1
		Soft	1	1	1
		Timer	1	1	1
	Timer		• 8-bit timer with 7-bit prescaler • Event counter		
	Clock Pulse Generator		• Resistor • Crystal		
	Low-voltage Automatic Reset (LVI)		Yes	Yes	Yes
	Self-check Mode		Available	Available	Available
	Other Features				
EPROM on the Package Type			–	–	HD68P05V07
Compatibility			MC6805P2	MC6805P6	–

* Preliminary

** Wide Temperature Range (−40 ~ +85°C) version is available.

HD6805V1	HD6805T2*	HD6805W1
1.0	1.0	1.0
5.25	5.25	5.25
0 ~ +70	0 ~ +70	0 ~ +70
DP-40	DP-28	DP-40
4	2.5	4
96	64	96
32 / 24 / 8	19 / 19 / –	29 / 23 / 6
6	6	12
1	1	2
1	1	1
1	1	4
		• 8-bit timer with 7-bit prescaler • Event counter • 8-bit comparator
	• Crystal	
Yes	Yes	Yes
Available	Available	Available
	PLL logic for RF synthesizer	• 8-bit x 4-channel internal A/D converter • 8 bytes of standby RAM
HD68P05V07	–	HD68P05W0
–	MC6805T2	–

■ CMOS 8-BIT SINGLE-CHIP MICROCOMPUTER HD6301 SERIES

Type No.			HD6301V1 HD63A01V1 HD63B01V1		HD6301X0 HD63A01X0 HD63B01X0	
LSI Characteristics	Bus Timing (MHz)		1.0 (HD6301V1) 1.5 (HD63A01V1) 2.0 (HD63B01V1)		1.0 (HD6301X0) 1.5 (HD63A01X0) 2.0 (HD63B01X0)	
	Supply Voltage (V)		5.0		5.0	
	Operating Temperature*** (°C)		0 ~ +70		0 ~ +70	
	Package		DP-40, FP-54, CG-40		DP-64S, FP-80	
Functions	Memory	ROM (k byte)	4		4	
		RAM (byte)	128		192	
	I/O Port	I/O Port	29	29	53	24
		Input Port		–		8
		Output Port		–		21
	Interrupt	External	2		3	
		Soft	2		2	
		Timer	3		4	
		Serial	1		1	
	Timer		16-bit x 1 (Free running counter x 1 Output compare register x 1 Input capture register x1)		16-bit x 1 (Free running counter x 1 Output compare register x2 Input capture register x1) 8-bit x 1 (8-bit up counter x 1 Time constant register x 1)	
	SCI		Asynchronous		Asynchronous/Synchronous	
	External Memory Expansion		65k bytes		65k bytes	
	Other Features		•Error detection •Low power consumption modes (sleep and standby)		•Error detection •Low power consumption modes (sleep and standby) •Slow memory interface •Halt	
EPROM on the Package Type (EPROM On-Chip Type)			HD63P01M1 HD63PA01M1* HD63PB01M1* HD63701V0†** HD637A01V0†** HD637B01V0†**		HD63701X0† HD637A01X0† HD637B01X0†	

* Preliminary ** Under development *** Wide Temperature Range (−40 ~ +85°C) version is available.
†EPROM on-chip type

HD6301Y0 HD63A01Y0 HD63B01Y0	HD6303R HD63A03R HD63B03R	HD6303X HD63A03X HD63B03X	HD6303Y HD63A03Y HD63B03Y
1.0 (HD6301Y0) 1.5 (HD63A01Y0) 2.0 (HD63B01Y0)	1.0 (HD6303R) 1.5 (HD63A03R) 2.0 (HD63B03R)	1.0 (HD6303X) 1.5 (HD63A03X) 2.0 (HD63B03X)	1.0 (HD6303Y) 1.5 (HD63A03Y) 2.0 (HD63B03Y)
5.0	5.0	5.0	5.0
0 ~ +70	0 ~ +70	0 ~ +70	0 ~ +70
DP-64S	DP-40, FP-54, CG-40	DP-64S, FP-80	DP-64S
16	–	–	–
256	128	192	256
53: 48 / – / 5	13: 13 / – / –	24: 16 / 8 / –	24: 24 / – / –
3	2	3	3
2	2	2	2
4	3	4	4
1	1	1	1
16-bit x 1 (Free running counter x1 Output compare register x 2 Input capture register x 1) 8-bit x 1 (8-bit up counter x 1 Time constant register x 1)	16-bit x 1 (Free running counter x 1 Output compare register x 1 Input capture register x 1)	16-bit x 1 (Free running counter x1 Output compare register x 2 Input capture register x 1) 8-bit x 1 (8-bit up counter x 1 Time constant register x 1)	16-bit x 1 (Free running counter x 1 Output compare register x 2 Input capture register x 1) 8-bit x 1 (8-bit up counter x 1 Time constant register x 1)
Asynchronous/Synchronous	Asynchronous	Asynchronous/Synchronous	Asynchronous/Synchronous
65k bytes	65k bytes	65k bytes	65k bytes
•Error detection •Low power consumption modes (sleep and standby) •Slow memory interface •Halt	•Error detection •Low power consumption modes (sleep and standby)	•Error detection •Low power consumption modes (sleep and standby) •Slow memory interface •Halt	•Error detection •Low power consumption modes (sleep and standby) •Slow memory interface •Halt

■ CMOS 8-BIT SINGLE-CHIP MICROCOMPUTER HD6305 SERIES

Type No.			HD6305U0* HD63A05U0* HD63B05U0*		HD6305V0* HD63A05V0* HD63B05V0*		HD6305X0 HD63A05X0 HD63B05X0	
LSI Characteristics	Clock Frequency (MHz)		1.0 (HD6305U0) 1.5 (HD63A05U0) 2.0 (HD63B05U0)		1.0 (HD6305V0) 1.5 (HD63A05V0) 2.0 (HD63B05V0)		1.0 (HD6305X0) 1.5 (HD63A05X0) 2.0 (HD63B05X0)	
	Supply Voltage (V)		5.0		5.0		5.0	
	Operating Temperature *** (°C)		0 ~ +70		0 ~ +70		0 ~ +70	
	Package		DP-40		DP-40		DP-64S, FP-64	
Functions	Memory	ROM (k byte)	2		4		4	
		RAM (byte)	128		192		128	
	I/O Port	I/O Port	31	31	31	31	55	32
		Input Port		—		—		7
		Output Port		—		—		16
	Interrupt	External	2		2		2	
		Soft	1		1		1	
		Timer	2		2		2	
		Serial	1		1		1	
	Timer							
	SCI							
	External Memory Expansion		—		—		—	
	Other Features							
EPROM on the Package Type (EPROM On-Chip Type)			—		HD63705V0†** HD637A05V0†** HD637B05V0†**		HD63P05Y0* HD63PA05Y0 HD63PB05Y0	
Evaluation Chip			—		—		—	

* Preliminary ** Under development *** Wide Temperature Range (−40 ~ +85°C) version is available.
† EPROM on-chip type

HD6305X1 HD63A05X1 HD63B05X1	HD6305X2 HD63A05X2 HD63B05X2	HD6305Y0 HD63A05Y0 HD63B05Y0	HD6305Y1 HD63A05Y1 HD63B05Y1	HD6305Y2 HD63A05Y2 HD63B05Y2	HD63L05F1
1.0 (HD6305X1) 1.5 (HD63A05X1) 2.0 (HD63B05X1)	1.0 (HD6305X2) 1.5 (HD63A05X2) 2.0 (HD63B05X2)	1.0 (HD6305Y0) 1.5 (HD63A05Y0) 2.0 (HD63B05Y0)	1.0 (HD6305Y1) 1.5 (HD63A05Y1) 2.0 (HD63B05Y1)	1.0 (HD6305Y2) 1.5 (HD63A05Y2) 2.0 (HD63B05Y2)	0.1
5.0	5.0	5.0	5.0	5.0	3.0
0 ~ +70	0 ~ +70	0 ~ +70	0 ~ +70	0 ~ +70	-20 ~ +75
DP-64S, FP-64	DP-64S, FP-64	DP-64S, FP-64	DP-64S, FP-64	DP-64S, FP-64	DP-64S, FP-80
4	–	8	8	–	4
128	128	256	256	256	96
31 / 24 / 7 / –	31 / 24 / 7 / –	55 / 32 / 7 / 16	31 / 24 / 7 / –	31 / 24 / 7 / –	20 / 20 / – / (19)
2	2	2	2	2	1
1	1	1	1	1	1
2	2	2	2	2	1
1	1	1	1	1	–
• 8-bit x 1 (with 7-bit prescaler) • 15-bit x 1 (combined with SCI)					• 8-bit x 1 (with 7-bit prescaler)
Synchronous					–
12k bytes	16k bytes	–	8 k bytes	16k bytes	–
• Low power consumption modes (Wait, stop and standby)					• 8-bit A/D converter • LCD driver (6 x 7 segment) • Low power consumption modes (Standby and halt)
HD63P05Y1* HD63PA05Y1* HD63PB05Y1*	–	HD63P05Y0 HD63PA05Y0 HD63PB05Y0	HD63P05Y1* HD63PA05Y1* HD63PB05Y1*	–	–
–	–	–	–	–	HD63L05E0

■ NMOS 8-BIT SINGLE-CHIP MICROCOMPUTER EPROM ON THE PACKAGE TYPE

Type No.		HD68P01V07	HD68P01V07-1	HD68P01M0	HD68P01M0-1	HD68P05V07	HD68P05M0
LSI Characteristics	Supply Voltage (V)	5.0				5.0	
	Operating Temperature (°C)	0 ~ +70				0 ~ +70	
	Package	DC-40P				DC-40P	
Equivalent Device		HD6801S0 HD6801V0	HD6801S5 HD6801V5	—	—	HD6805U1 HD6805V1	—
Mountable EPROM		HN482732A-30	HN482732A-30	HN482764-3	HN482764-3	HN482732A-30	HN482764-3

■ CMOS 8-BIT SINGLE-CHIP MICROCOMPUTER EPROM ON THE PACKAGE TYPE

Type No.		HD63P01M1	HD63PA01M1*	HD63PB01M1*	HD63P05Y0	HD63PA05Y0	HD63PB05Y0
LSI Characteristics	Supply Voltage (V)	5.0			5.0		
	Operating Temperature (°C)	0 ~ +70			0 ~ +70		
	Package	DC-40P			DC-64SP		
Equivalent Device		HD6301V1	HD63A01V1	HD63B01V1	HD6305X0 HD6305Y0	HD63A05X0 HD63A05Y0	HD63B05X0 HD63B05Y0
Mountable EPROM		HN482732A-30 HN482764-3 HN27C64-30	HN482732A-30 HN482764-3 HN27C64-30	HN482732A-25 HN482764 HN27C64-25	HN482732A-30 HN482764-3 HN27C64-30	HN482732A-30 HN482764-3 HN27C64-30	HN482732A-25 HN482764 HN27C64-25

* Preliminary

■ CMOS 8-BIT SINGLE-CHIP MICROCOMPUTER EPROM ON-CHIP TYPE

Type No.			HD6370IV0*	HD63701X0	HD63705V0*
LSI Characteristics	Clock Frequency (MHz)		1.0, 1.5, 2.0	1.0, 1.5, 2.0	1.0, 1.5, 2.0
	Supply Voltage (V)		5.0	5.0	5.0
	Package		DC-40 DP-40	DC-64S DP-64S	DC-40 DP-40
Functions	Memory	EPROM (k byte)	4	4	4
		RAM (byte)	192	192	192
	I/O Port	I/O Port	29	53	31
		Input Port	—	8	—
		Output Port	—	21	—
	Interrupt	External	2	3	2
		Soft	2	2	1
		Timer	3	4	2
		Serial	1	1	1
	Timer		16 bit x 1	16 bit x 1 8 bit x 1	8 bit x 1
	Serial I/O		UART	UART synchronous	Synchronous
	External Memory Expansion		Possible (65k byte)	Possible (65k byte)	—
	Others		• Error detection • Low power consumption modes (Sleep and Standby)	• Error detection • Low power consumption modes (Sleep and Standby) • Slow memory interface	• Low power consumption modes (Wait, Stop and Standby)
EPROM Programming	Equivalent EPROM Type		27C256, 27256 (V_{PP} = 12.5V) High Performance Programming algorithm available	2732A (V_{PP} = 21V)	27C256, 27256 (V_{PP} = 12.5V) High Performance Programming algorithm available
	Socket Adapter	Hitachi	(under development)	H67PWA01A H67PWA01B	H35VSA00A H35VSA00B
		Data I/O	—	HD63701X0 (for 29A/29B)	HD63705V (for 29A/29B)

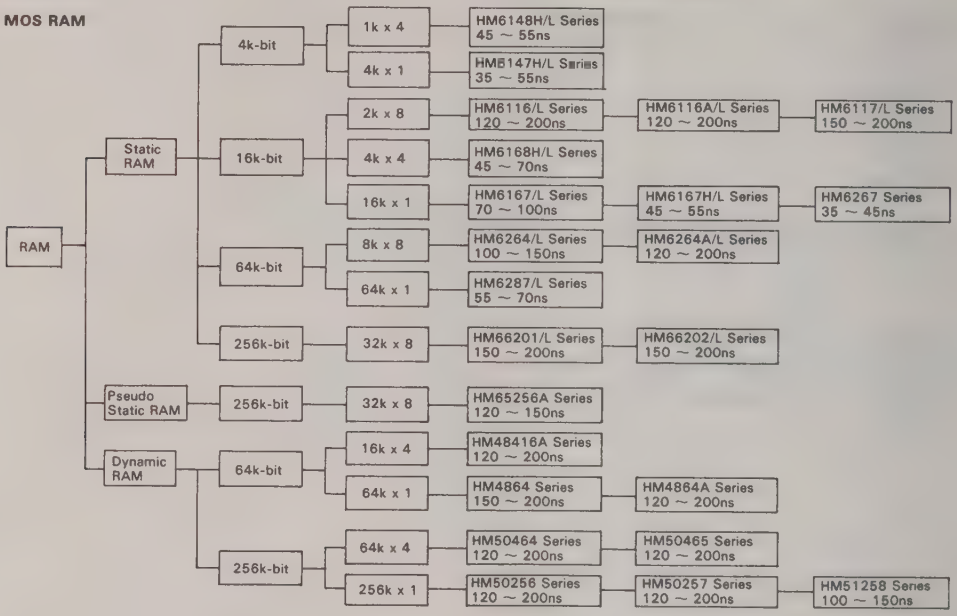
* Under development

HD68P05W0
5.0
0 ~ +70
DC-40P
HD6805W1
HN482732A-30 HN482764-3

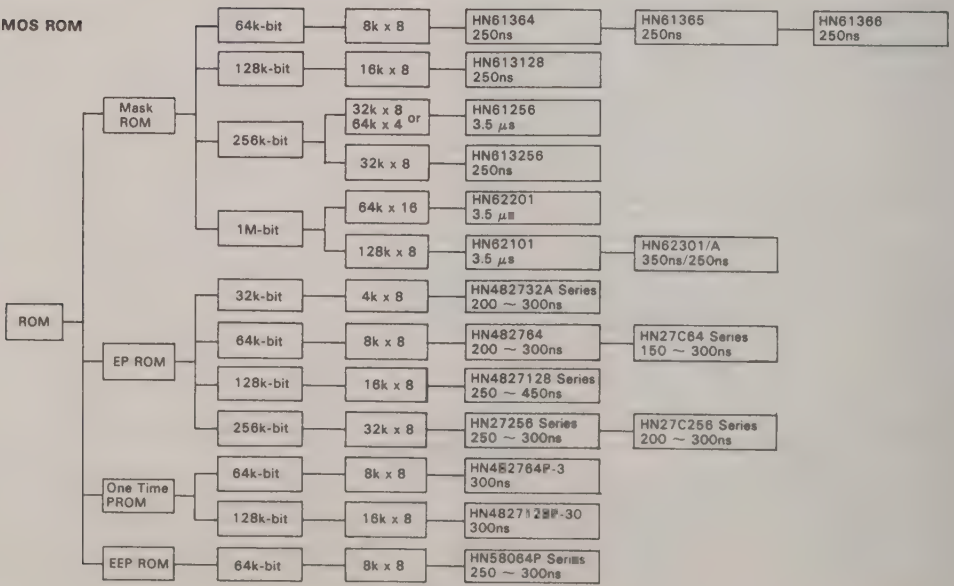
HD63P05Y1*	HD63PA05Y1*	HD63PB05Y1*
5.0		
0 ~ +70		
DC-64SP		
HD6305X1 HD6305Y1	HD63A05X1 HD63A05Y1	HD63B05X1 HD63B05Y1
HN482732A-30 HN482764-3 HN27C64-30	HN482732A-30 HN482764-3 HN27C64-30	HN482732A-25 HN482764 HN27C64-25

IC MEMORY

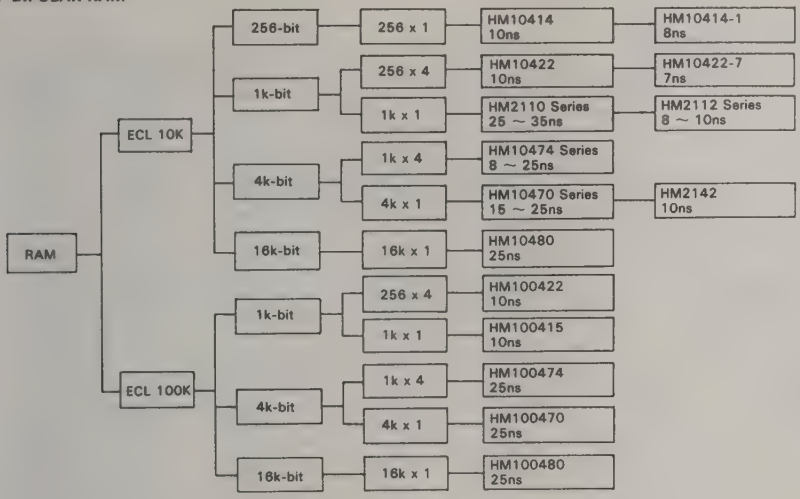
■ MOS RAM



■ MOS ROM



■ **BIPOLAR RAM**


RAM
ECL 10K
256-bit
256 x 1
HM10414
10ns
HM10414-1
8ns
1k-bit
256 x 4
HM10422
10ns
HM10422-7
7ns
1k x 1
HM2110 Series
25 ~ 35ns
HM2112 Series
8 ~ 10ns
4k-bit
1k x 4
HM10474 Series
8 ~ 25ns
4k x 1
HM10470 Series
15 ~ 25ns
HM2142
10ns
16k-bit
16k x 1
HM10480
25ns
ECL 100K
1k-bit
256 x 4
HM100422
10ns
1k x 1
HM100415
10ns
4k-bit
1k x 4
HM100474
25ns
4k x 1
HM100470
25ns
16k-bit
16k x 1
HM100480
25ns

■ LCD DRIVER SERIES CHARACTERISTICS

Item \ Use			General			Segment Display	
LSI Characteristics	Type Number		HD44100H	HD61100A	HD61200	HD61602	HD61603
	Process		CMOS	CMOS	CMOS	CMOS	CMOS
	Supply Voltage	(V)	5*1	5*1	5*1	3 ~ 5*1	3 ~ 5*1
	Operating Temperature	(°C)	-20 ~ +75*2	-20 ~ +75	-20 ~ +75	-20 ~ +75	-20 ~ +75
	Package		FP-60	FP-100	FP-100	FP-80	FP-80
	Power Dissipation	(mW)	5.0	5.0	5.0	0.5 (5V)	0.5 (5V)
Functions	Memory — ROM	(bits)	–	–	–	–	–
	Memory — RAM	(bits)	–	–	–	51 x 4	64 x 1
	I/O — Interface (CPU)		8	8	8	14	10
	I/O — Interface (Driver IC)		2	2	2	–	–
	I/O — Interface (External ROM, RAM)		–	–	–	–	–
	Number of Instruction		–	–	–	4	4
	LCD Driver — Common		40	80	80	4	1
	LCD Driver — Segment					51	64
	LCD Driver — Duty		Free (N)	Free (N)	Free (N)	Static, 1/2, 1/3, 1/4	Static
	Display Capability		N x 40 Matrix (1/N Duty)	N x 80 Matrix (1/N Duty)	N x 80 Matrix (1/N Duty)	204 Segment (1/4 Duty)	64 Segment
Comment			SR type	SR type	SR type		

Item \ Use			Graphic Display				
LSI Characteristics	Type Number		HD44102CH	HD44103CH	HD44105H	HD61102	HD61202
	Process		CMOS	CMOS	CMOS	CMOS	CMOS
	Supply Voltage	(V)	5*1	5*1	5*1	5*1	5*1
	Operating Temperature	(°C)	-20 ~ +75	-20 ~ +75	-20 ~ +75	-20 ~ +75	-20 ~ +75
	Package		FP-80	FP-60	FP-60	FP-100	FP-100
	Power Dissipation	(mW)	2.5	4.0	4.0	3.0	2.5
Functions	Memory — ROM	(bits)	–	–	–	–	–
	Memory — RAM	(bits)	200 x 8	–	–	512 x 8	512 x 8
	I/O — Interface (CPU)		21	6	6	21	21
	I/O — Interface (Driver IC)		–	5	5	–	–
	I/O — Interface (External ROM, RAM)		–	–	–	–	–
	Number of Instruction		6	–	–	7	7
	LCD Driver — Common		–	20	32	–	–
	LCD Driver — Segment		50	–	–	64	64
	LCD Driver — Duty		1/8, 1/12, 1/16, 1/24, 1/32	1/8, 1/12, 1/16, 1/24, 1/32	1/8,1/12,1/16,1/24, 1/32, 1/48, 1/64	1/64	1/64
	Display Capability		32 x 50 Dots (1/32 Duty)	–	–	64 x 64 Dots (1/64 duty)	64 x 64 Dots (1/64 duty)
Comment			Suitable common driver is HD44105H or HD44103CH			Suitable common driver is HD61103A	Suitable common driver is HD61203

*1: Except Power Supply for LCD.
*2: -40 ~ +85°C (Special Request). Please contact Hitachi Agents.
*3: CG; Character Generator.

Character Display	
HD44780 (LCD-II)	HD43160AH
CMOS	CMOS
5*1	5*1
-20 ~ +75*2	-20 ~ +75
FP-80	FP-54
1.75	10.0
7200 (CG)*3	6240 (CG)*3
80 x 8/64 x 8 (CG)*3	80 x 8
11	21
4	5
–	18
11	6
16	–
40	–
1/8, 1/11, 1/16	1/8, 1/12, 1/16
16 Digits (5 x7 Dots 1/16 Duty)	–
Expandable to 80 Digits using HD44100H	Display to 80 Digits using HD44100H

Graphic Display			
HD61103A	HD61203	HD61830	HD618308
CMOS	CMOS	CMOS	CMOS
5*1	5*1	5*1	5*1
-20 ~ +75	-20 ~ +75	-20 ~ +75	-20 ~ +75
FP-100	FP-100	FP-60	FP-60
5.0	5.0	30.0	30.0
–	–	7360 (CG)*3	7360 (CG)*3
–	–	(external 65536 x 8)	(external 65536 x 8)
6	6	13	13
5	5	9	9
–	–	33	33
–	–	12	12
64	64	–	–
–	–	–	–
1/48, 1/64, 1/96, 1/128	1/48, 1/64, 1/96, 1/128	1/1 ~ 1/128	Static 1/1 ~ 1/128
–	–	–	–
		Display to 524288 Dots using HD44100H or HD61100A	

GATE ARRAY

CMOS Gate Array HD61J/HD61K/HD61L/HD61MM Series

■ FEATURES

- Fast operation
  Internal gate (2-input NAND, FO=3, AL=3mm) . . 3.5ns typ
  Input buffer (FO=3, AL = 3mm) . . . . . 9ns typ
  Output buffer (C_L=50pF) . . . . . 20ns typ
  Memory access time (HD61MM) . . . . . 60ns typ
- Low power dissipation
  At 10MHz operation (Internal gate) . . . . . 130μW/gate typ
- Abundant input and output configuration
  Allocation of all pins except power supply pins to input/output/input-output
  Output can be CMOS/open drain/3-state
- Memory on-chip (HD61MM)
  Flexibility of memory capacity and word organization
  Selection of single port/dual port memory
- Wide operation temperature range
  −20 to +75°C
- Wide package selection
  Especially plastic packages with high pin count DILP64/FPP100
- Powerful design support
  User-Defined-Macro
  Test pattern evaluation with fault simulator
  Design support at local Design Center
- Quick turn around time and reasonable development cost

■ LINE UP

		HD61J	HD61K	HD61L	HD61MM
Gate count		504	1080	1584	2496
I/O pin count		50	68	68	104
RAM on chip		—	—	—	available
Package	DP28	○	○	○	—
	DP40	*	*	○	*
	DP42	○	○	○	—
	DP64	—	○	○	○
	FP54	○	—	—	—
	FP80	—	○	○	—
	FP100	—	—	—	*
	DC28	○	○	○	*
	DC40	○	○	○	*
	PGA72	—	—	○	—
	PGA120	—	—	—	*
Power supply pin		*	4		4 8*

*Under development

Bi-CMOS Gate Array HD27K/HD27L/HD27P/HD27Q Series

■ FEATURES

- High speed with super low power dissipation . . . . . .
  - Internal gate: 4.0ns (Fan out=3) @0.05mW
  - Input buffer: 5.0ns (Fan out=3) @2.6mW
  - Output buffer: 8.0ns (C_L=15pF) @2.6mW
- LS TTL compatible input/output . . . . . .
  - Selectable totem-pole/3-state/open collector output
  - I_{OL}=8mA: Capable of driving 20 LS TTL's
- Output buffer can construct logic functions.
  - Saves gate stages.
- A variety of macrocell library
  - Internal gate: 44
  - Output buffer: 9
- A variety of reliable package
  - Plastic DIP 16 to 64 pins
  - Plastic FP 60 to 100 pins (under development)
- A variety of DA system support
  - Only logic diagrams and test patterns needed as an interface with the user.
- Short development time

	Number of gates			Number of V_{CC} and GND pins	Package	
	Internal gate (2-input NAND)	Input buffer	Output buffer		DIP (Plastic)	FP* (Plastic)
HD27K	200	18	18	2	16, 20, 28, 40 pins	–
HD27L	528	30	30	4	28, 42, 64 pins	60 pins
HD27P	966	40	40	4	28, 42, 64 pins	60, 80, 100 pins
HD27Q	1530	50	50	4	28, 42, 64 pins	60, 80, 100 pins

*Under development

CODEC/FILTER COMBO LSI

SERIES	TYPE	COMP. LAW	POWER (Typ.)	CLOCK			DECODER SHIFT	INPUT AMP	OUTPUT AMP	
				INTERNAL CLOCK	SYNC/ASYNC OPERATION	PCMBIT CLOCK RATE			TYPE	MIN LOAD
44230	HD44231B	A	60mW	DEVIDER INCLUDED	SYNC. ONLY	1536/ 1544/ 2048kHz	–	ADJUST-ABLE USING 2 RESIST	SINGLE ENDED	3 kΩ
	HD44232B	μ								
	HD44233B	A			BOTH					1.2 kΩ
	HD44234B	μ								
	HD44235	A	50mW	PLL INCLUDED	SYNC. ONLY	64– 2048kHz	–			
	HD44236	μ					○			
	HD44237	A			BOTH		–			
	HD44238	μ					○			
44230C	HD44231C	A	60mW	DEVIDER INCLUDED	SYNC. ONLY	1536/ 1544/ 2048kHz	–	FULLY UNCOMM-ITED OP-AMP		600 Ω
	HD44232C	μ								
	HD44233C	A			BOTH					
	HD44234C	μ								
	HD44235C	A		PLL INCLUDED	SYNC. ONLY	64– 2048kHz	–			
	HD44236C	μ					○			
	HD44237C	A			BOTH		–			
	HD44238C	μ					○			
44240C	HD44247C	A	80mW	PLL INCLUDED	BOTH	64– 2048kHz	–	SAME AS ABOVE	PUSH-PULL	600 Ω
	HD44248C	μ					○			

SPEECH SYNTHESIS LSI

Type	CMOS 1-chip System
Device	HD61885/7 (Speech Synthesizer) HD44881 (128k-bit Expanding ROM)
System	PARCOR
Voice channel model	10 steps digital filter
Sampling frequency	10 kHz
Bit rate (b/s)	1250 ~ 9900
Frame period (ms)	10/20
Variable speaking speed	–25%, 0, +25%
Speaking time	10 ~ 20 sec (internal ROM)

HITACHI AMERICA, LTD.
SEMICONDUCTOR AND IC DIVISION

HEADQUARTERS

Hitachi, Ltd.
Nippon Bldg., 6-2, 2-chome
Ohtemachi, Chiyoda-ku, Tokyo, 100, Japan
Tel: 212-1111
Telex: J22395, J22432

U.S. SALES OFFICE

Hitachi America, Ltd.
Semiconductor and IC Division
2210 O'Toole Avenue
San Jose, CA 95131
Tel: 408-942-1500
Telex: 17-1581
Twx: 910-338-2103
Fax: 408-942-8225
Fax: 408-942-8880

REGIONAL OFFICES

NORTHEAST REGION

Hitachi America, Ltd.
5 Burlington Woods Drive
Burlington, MA 01803
617/229-2150

SOUTH CENTRAL REGION

Hitachi America, Ltd.
Two Lincoln Centre, Suite 865
5420 LBJ Freeway
Dallas, TX 75240
214/991-4510

NORTH CENTRAL REGION

Hitachi America, Ltd.
500 Park Blvd., Suite 415
Itasca, IL 60143
312/773-4864

NORTHWEST REGION

Hitachi America, Ltd.
2099 Gateway Place, Suite 550
San Jose, CA 95110
408/277-0712

SOUTHWEST REGION

Hitachi America, Ltd.
21600 Oxnard St., Suite 600
Woodland Hills, CA 91367
818/704-6500

SOUTHEAST REGION

Hitachi America, Ltd.
4901 N.W. 17th Way, Suite 302
Fort Lauderdale, FL 33309
305/491-6154

DISTRICT OFFICES

- Hitachi America, Ltd.
  1700 Galloping Hill Rd.
  Kenilworth, NJ 07033
  201/245-6400
- Hitachi America, Ltd.
  3500 W. 80th Street, Suite 660
  Bloomington, MN 55431
  612/831-0408
- Hitachi America, Ltd.
  80 Washington St., Suite 302
  Poughkeepsie, NY 12601
  914/485-3400
- Hitachi America, Ltd.
  6 Parklane Blvd., #558
  Dearborn, Mi 48126
  313/271-4410
- Hitachi America, Ltd.
  6161 Savoy Dr., Suite 850
  Houston, TX 77036
  713/974-0534
- Hitachi America, Ltd.
  5775 Peachtree-Dunwoody Rd.
  Suite 270C
  Atlanta, GA 30342
  404/843-3445
- Hitachi America, Ltd.
  18300 Von Karman Avenue, Suite 730
  Irvine, CA 92715
  714/553-8500
- Hitachi America, Ltd.
  10300 S.W. Greenburg Rd., Suite 480
  Portland, OR 97223
  503/245-1825
- Hitachi (Canadian) Ltd.
  2625 Queensview Dr.
  Ottawa, Ontario, Canada K2A 3Y4
  613/596-2777

HITACHI

A World Leader in Technology

Hitachi America, Ltd.
Semiconductor and IC Division
2210 O'Toole Avenue, San Jose, CA 95131
1-408-942-1500

10M

Printed in U.S

www.manualsplus.com

1027488